Courant Computer Science Report #2

January 1974

A Metalanguage for Expressing Grammatical Restrictions in Nodal Spans Parsing of Natural Language

Jerry R. Hobbs

Courant Institute of
Mathematical Sciences

Computer Science Department

New York University

Report No. NSO-2 prepared under Grant No.
NSF-GJ-1202x2 from the National Science Foundation

C.2

NSO-2

COURANT COMPUTER SCIENCE PUBLICATIONS

COURANT COMPUTER SCIENCE NOTES

	Price
On Programming; An Interim Report on the SETL Project. Installment I: Generalities J. T. Schwartz, *1973, vii+160 pp.*	$ 4.25
Installment II: The SETL Language and Examples of Its Use J. T. Schwartz, *1973, viii+520 pp.*	13.00
Compilation and Optimization of Compiled Code (formerly titled: Programming Languages & Their Compilers), J. Cocke and J. T. Schwartz, 2nd Rev. Version, *April 1970, iii+767 pp.*	12.50
A SETLB Primer, H. Mullish and M. Goldstein, *1973, v+201 pp.*	5.25
Combinatorial Algorithms, E. G. Whitehead, Jr., *1973, vi+104 pp.*	2.75

COURANT COMPUTER SCIENCE REPORTS

No. 1 ASL: A Proposed Variant of SETL
Henry Warren, Jr., *1973, 326 pp.*

No. 2 A Metalanguage for Expressing Grammatical Restrictions in Nodal Spans Parsing of Natural Language, Jerry R. Hobbs, *1974, 266 pp.*

A catalog of SETL Newsletters and other SETL-related material is also available. Courant Computer Science Reports are available upon request. Prepayment is required for Courant Computer Science Notes. Please address all communications to

COURANT INSTITUTE OF MATHEMATICAL SCIENCES
251 Mercer Street
New York, N. Y. 10012

COURANT INSTITUTE OF MATHEMATICAL SCIENCES

Computer Science NSO-2

A METALANGUAGE FOR EXPRESSING GRAMMATICAL RESTRICTIONS IN NODAL SPANS PARSING OF NATURAL LANGUAGE

Jerry R. Hobbs

Report No. NSO-2 prepared under
Grant No. NSF-GJ-1202X2 from
the National Science Foundation

TABLE OF CONTENTS

Chapter

1. Introduction 1

1.1. String Analysis 1

1.2. Automatic Parsing of Natural Language 8

1.3. Parsing Algorithms 13

1.4. Ambiguity in Natural Language 16

2. The Grammar 24

2.1. The Components of the Grammar 24

2.2. The BNF Component 25

2.3. The Restrictions 45

2.4. Locating Relations 51

2.5. The Word Dictionary 54

3. The Nodal Spans Algorithm 56

3.1. Informal Description 56

3.2. Formal Description 57

3.3. Example 62

3.4. Candidates Lists 67

3.5. Space and Time Requirements 69

3.6. History 72

4. Nodal Spans Parsing without Restrictions: Statistical Results 73

4.1. Introduction 73

4.2. The Inverse Grammar 76

4.3. The Nonmerging Parsing Algorithm 78

4.4. The Merging Parsing Algorithm 83

4.5. Results 87

Chapter

5. A Metalanguage for Expressing Grammatical Restrictions 100

5.1. Introduction and Terminology 100

5.2. General Description of the Restriction Metalanguage; The Core 103

5.3. Housing Specifications 109

5.4. Flag- and Attribute-Checking Statements 111

5.5. Forming the New Core; Flag-Setting Instructions 114

5.6. Endpoint Checking Statements; Comma Restrictions 117

5.7. Span Checking Statements 118

5.8. Semigrammaticality Markers 119

5.9. Syntax of the Restriction Metalanguage 120

5.10. Semantics of the Restriction Metalanguage 123

5.11. The Language is Still Context-Free 129

6. The Metalanguage Applied to the Sager Grammar for English 133

6.1. Introduction 133

6.2. Restructuring the Assertion String 135

6.3. Restructuring the Left Adjunct of the Noun 138

6.4. Adverbial Preposition - Preposition Strings 141

6.5. OBJLIST Restrictions 143

6.6. Writing Restrictions 146

6.7. Examples of Restrictions in Action 152

6.8. The Complete Algorithm 164

6.9. The Implementation 170

Chapter

7. An Algorithmic Approach to Semigrammaticality 173

7.1. Introduction 173

7.2. The Algorithm 176

7.3. The Hierarchy 180

Bibliography 189

Appendix

A. String Grammar of English for Bottom-Up Parsing 193

Index of Nonterminal Symbols 203

B. The Restrictions Expressed in the Bottom-Up Restriction Metalanguage. 206

Deepest Verb and Deepest Object of "Be" Restrictions. 206

Agreement Restrictions 208

Comma Restrictions 211

Noun Phrase Restrictions 214

Positional Restrictions 219

Quantities Restrictions 229

Selectional Restrictions 231

Sentence Nominalization Restrictions 232

Verb and Center Restrictions 241

Wh-String Restrictions. 243

C. Index of Flags 252

D. String Grammar of English used in Statistical Studies of Chapter 4. 260

Abstract

Within the level of syntactic analysis by computer possible today, a large number of English sentences are ambiguous, even highly ambiguous. In a natural language processing system which is too large for disambiguation by a semantic component, it may often be desirable to produce all parses of a setnence. Top-down single-tracking algorithms necessarily require exponential time in this environment. The bottom-up nodal spans, or Earley's parsing algorithm procuces all parses in parallel in time n^3, yields a convenient representation of ambiguity, and therefore seems more desirable for this task. Briefly, if a portion of an input string is capable of two different interpretations which in turn can be subsumed under the same symbol, the algorithm represents this symbol by only one node in the tree-like structure it builds. Ambiguity is thus localized and prevented from causing exponential explosion. In addition, a device known as candidates lists, or predictor, constrains possible interpretations to those consistent with the interpretations of earlier parts of the input. A disadvantage of the algorithm, however, is that it also requires n^3 space.

This dissertation confronts the problem of adapting the nodal spans algorithm to the parsing of natural language. The grammar taken as a prototype is the reasonably broad and detailed surface grammar of English developed by Naomi Sager at the Linguistic String Project at New York University. This grammar consists of a context-free component of about 200 productions which indicate the possible constructions of English, and a set of about 200 restrictions which look at attributes of words in the input to check the parse tree as it is evolved for well-formedness.

First, statistical tests were run to determine if the algorithm could be applied to the context-free component alone, with the restrictions being applied afterwards. Disastrous explosions in space requirements resulted. Nevertheless, comparative tests indicated the form the algorithm would have to take in subsequent development. In particular, candidates lists were found to save up to 50% in space and time. Results of these tests are presented.

Nodal spans parsing of English therefore requires the expression of restrictions in a form appropriate to the algorithm. In particular, they must act strictly locally. Because different interpretations of an ambiguous portion of a sentence may be merged under one span, it is not possible to traverse the parse tree easily; information needed by a restriction must be anticipated and made readily available at the point at which the restriction is applied.

A metalanguage for expressing restrictions is developed. It is based upon associating with each node in the parse tree a word in the input string as its core and a set of flags which signal structural features in the subtrees below the node. Restrictions are expressed in terms of these flags and the attributes of this core word, and can be executed quickly without time-consuming tree traversals. This metalanguage is described and defined and applied to the entire restriction component of the Sager grammar. Interesting problems that come up along the way are discussed. The system is currently being implemented at New York University.

The nodal spans algorithm and the metalanguage approach to restrictions lend themselves to an algorithmic treatment of semigrammaticality, and this is also developed.

CHAPTER I. INTRODUCTION

This work confronts the problem of adapting the fast bottom-up parallel parsing technique known as nodal spans to the syntactic analysis of natural language. After a brief history and discussion of the issues involved, an account is given of the machine grammar of English developed by Naomi Sager as an example of the type of broad and detailed grammars necessary in natural language processing. Then the nodal spans algorithm is described. To demonstrate the necessity of expressing grammatical restrictions in a form appropriate to the algorithm, statistical tests are run to determine the feasibility of parsing without such restrictions. A metalanguage is described and defined for the expression of grammatical restrictions. This is considered the chief outcome of the work. To demonstrate the usefulness and power of the metalanguage, its application to the Sager grammar is presented.

The work described here results from the convergence of three developments which are outlined in turn below.

1.1. String Analysis

The first development begins with the increased interest of linguists in the syntactic analysis of sentences in the last forty years. Leonard Bloomfield[1] mentioned that sentences

[1] Leonard Bloomfield, Language, New York, 1933, p. 161.

could be analyzed into immediate constituents. For example, the sentence

"Man uses language."

can be broken into a subject "man" and a predicate "uses language" which can in turn be broken into the verb "uses" and an object "language". Zellig Harris[2] outlined a technique for the discovery of the grammatical roles of sequences of words or morphemes in sentences, viz. substitution of the sequence of words for a single word, and gave a sample list of substitution rules for English; he also remarked that the same rules could be used for immediate constituent analysis. Rulon Wells[3] expanded this remark into what was for the time a reasonably detailed immediate constituent analysis of English.

In the early 1950's the importance of grammatical transformations in syntactic analysis was recognized. Zellig Harris[4] first mentions them as an aid in discourse analysis; in this article transformations are seen primarily as equivalence relations between sentences. Two sentences are transformationally related if one can be transformed into the other by the rearrangement and slight alterations of its set of morphemes and if the resulting sentence can appear in the

---

[2] Zellig Harris, "From Morpheme to Utterance," Language, Vol. 22, p. 165 - 183, 1946.

[3] Rulon S. Wells, "Immediate Constituents," Language, Vol. 23, pp. 81-117, 1947.

[4] Zellig Harris, "Discourse Analysis," Language, Vol. 28, pp. 1-30, 1952.

same contexts as the original. Thus, the sentences

"Man uses language."

and "Language is used by man."

are related transformationally. In the middle and late 1950's, especially in the work of Noam Chomsky,[5] the study of transformations became central; he proposed a model for language in which a base component produces kernel trees or sentences which are then operated upon by transformations to produce the full variety of natural language sentences. Thus, if the two sentences

"Men conceal the truth."

and "Men abuse language."

were produced by the base component and acted upon by "passive" and embedding transformations, the sentence

"The truth is concealed by men who abuse language."

would be produced.

While many linguists have concerned themselves with the generation of grammatical sentences, the inverse problem is clearly also important. Given a grammatical sentence, how can one determine which transformation produced it and what kernel sentences underlie it? Harris'[6] development of the method of string analysis provides the beginning of an answer to that question.

String analysis, like immediate constituent analysis, has as its goal the resolution of the sentence into a

---

[5] Noam Chomsky, Syntactic Structures, The Hague, 1957.

[6] Zellig Harris, String Analysis of Sentence Structure, The Hague, 1962.

tree-like structure. In string analysis the principal constituents of a sentence are Subject - Verb - Object:

"Man uses language."

for transitive verbs, or for intransitive verbs, simply Subject - Verb:

"Language exists."

String analysis also recognizes the importance of other elements which are adjoined to the basic sentence at various points to modify or amplify what is said. One type of adjunct, the "sentence adjunct," acts at the same level as the principal constituents and serves to modify the entire sentence or relate it to other sentences in the discourse:

"As a result, language exists."

"In addition, man uses language."

Another type of adjunct is adjoined to the left or right of specific elements and serves to modify only those elements:

"That chubby man with the cigar and the bowler hat frequently uses the English language."

The adjuncts are optional in that deletion of them would result, for the most part, only in the subtraction of detail, and not in the loss or alteration of meaning. Furthermore, after deletion of these adjuncts the sentences would with certain exceptions still be grammatical.

Each of the principal constituents has as its "core" a word or string to which the adjuncts are attached. In the sentence

"His argument is confusing." (1)

the core of the subject is the word "argument" to which the adjunct "his" is attached. A string is the resultant of a transformation applied to an assertion, and under certain restrictions can replace a word and its adjuncts as a principal constituent. For example, a nominalizing transformation applied to the assertion

"He says something."

yields the string

"what he says"

which can be substituted for the subject in sentence (1) --

"What he says is confusing."

Another nominalizing transformation applied to the sentence

"He means the opposite of what he says."

yields the string

"his meaning the opposite of what he says"

which can also be used as the subject of sentence (1) --

"His meaning the opposite of what he says is confusing." (2)

This device of substitution is first of all a logical extension of the classical method of discovery and analysis in descriptive linguistics and grows naturally out of Harris' earlier work on morpheme sequences.[7] Second of all, substitution is one of the devices by which man is able, in Chomsky's words, to get infinite use out of the finite resources of language. Three kernel sentences underly example (2); theoretically there is no limit on the number of kernel sentences which may underly a sentence.

---

[7] Zellig Harris, "From Morpheme to Utterance," Language, Vol. 22, p. 161, 1946.

String analysis, viewing sentences as made up of principal constituents and optionally adjoined elements, lends itself to another informal method for the discovery and analysis of grammars, viz. excision. Suppose we are given a collection of utterances in a new language which have previously been divided into words or morphemes, and suppose we have an informant who will tell us when we excise a portion of a sentence

a. whether the remainder still means the same, or whether we have subtracted detail, or whether we have changed the meaning entirely, and

b. whether the remainder is still a grammatical sentence.

Consider for example the sentence

"There is a compass on the house boat."

If we excise "house",

"There is a compass on the boat."

our informant will tell us we have merely subtracted detail. If we excise "boat",

"There is a compass on the house."

he will say we still have a grammatical sentence but its meaning has been radically altered. These examples lead us to postulate that the second noun is the critical element and the first is an adjunct. Next we excise "the",

"There is a compass on house boat."

Now he tells us that in a sense we still have the same meaning, but we have lost grammaticality. This induces us

to propose a restriction that words of a certain category require words like "the" to precede them. Looking for larger elements we excise the phrase "on the house boat"

"There is a compass."

and our informant says we have subtracted detail. If we excise "a compass",

"There is on the house boat."

he tells us that we have lost grammaticality and changed it into a nonsense utterance. These trials lead us to say that "on the house boat" is an adjunct while "a compass" is a crucial part of the sentence. Then we excise "there"

"Is a compass on the house boat."

He now tells us that we have a grammatical sentence whose meaning, however, has changed. But the meaning is not very different, and this fact may lead us to deduce a transformation.

The method of excision may also be used to provide evidence for making decisions about the grammar. Suppose for example we must decide for the phrase

"his wearing of Superman suits"

whether it is a string derived from the kernel sentence

"He wears Superman suits."

or whether it is merely the gerund "wearing" flanked by the left adjunct "his" and the right adjunct "of Superman suits". If "of Superman suits" is merely an adjunct, it is optional and can be excised. However, when we excise it, we obtain

"his wearing"

which lacks meaning. We therefore conclude that the phrase is in fact a string.

The goal of string analysis is the recognition of the strings in a sentence. It therefore occupies an intermediate position between immediate constituent analysis and transformational analysis. Immediate constituent analysis offers no help in detecting the action of transformations. By contrast, it is likely that once the strings in a sentence are recognized, one is in a very good position indeed to determine what transformations took part in producing the sentence.

1.2. Automatic Parsing of Natural Languages

Spurred in part by the failure of early attempts in mechanical translation, interest grew in the automatic parsing of natural language in the early 1960's. The chief problem in this regard is constructing a grammar for English large enough to encompass the vast variety of syntactic constructions and detailed enough to eliminate spurious ambiguities.

One of the earliest major efforts was the Harvard Predictive Analyzer developed by Kuno and Oettinger.[8] Their goal was to construct automatically trees reflecting the

[8] Susumu Kuno and Anthony G. Oettinger, "Syntactic Structure and Ambiguity of English," Proceedings -- Fall Joint Computer Conference, 1963, pp. 397-418.

surface structure of English sentences. For this purpose, a very large purely context-free grammar was developed. The Harvard Predictive Analyzer went through a number of changes, but at one point had approximately 2100 productions and about 130 terminal symbols defining word subclasses. With all of this machinery, the number of false ambiguities in the analyses the parser produced was still considered unacceptable. The two most important conclusions that could be drawn from Kuno and Oettinger's work were first that the problem of syntactic analysis of English is enormous, and second that the grammar cannot be expressed as a large purely context-free grammar. One would like to require reasonable brevity and comprehensibility of the grammar and a grammar of 2100 productions already fails on these counts. One would also like to be able to extend the grammar easily. In a large context-free grammar adding a new grammatical constraint may entail adding only a few more word subclasses, but if their usage depends upon distant elements in the sentence, it may be necessary to add a large number of new productions and major difficulties may be caused with current productions.

The syntactic analyzer PARSE developed at RAND by Jane Robinson[9] was also based on a purely context-free grammar and also sought to analyze the surface structure of English

[9] Jane Robinson and Shirley Marks, PARSE: A System for Automatic Syntactic Analysis of English Text, Rand Corp., 1965.

sentences. It contained approximately 3500 productions and a large number of terminal symbols representing word subclasses. Tests were run on a number of sentences. Although the parser gave reasonable results on many sentences, there were several examples of very high degrees of ambiguity -- a 13-word sentence with 35 parses; an 18-word sentence with 91 parses; a 27-word sentence with 106 parses; and a 33-word sentence with over 180 parses. The conclusions drawn from Kuno and Oettinger's work apply here as well; the analysis was not detailed enough and the grammar was not in a form that made it easy to expand.

A very different approach was taken at the MITRE Corporation by Arnold Zwicky and others.[10] Their goal was to recognize the transformations that underlie English sentences. The analysis proceeded in four stages:

1. The English sentence was parsed according to a surface grammar to produce a tree representing the surface structure of the sentence. The surface grammar consisted of approximately 550 context-free productions.

2. Using the output of the first stage, reverse transformations were applied to derive sets of presumed basic trees of kernel sentences. There were about 134 reverse transformations.

---

[10] Arnold Zwicky, Joyce Friedman, Barbara C. Hall, and Donald Walker, "The MITRE Syntactic Analysis Procedure for Transformational Grammars," Proceedings -- Fall Joint Computer Conference, 1965, pp. 317-326.

3. Some of these presumed basic trees were inconsistent with the base component of the grammar. Therefore the basic trees which could not be generated by the base component were discarded.

4. Forward transformations were applied to the remaining sets of basic trees to determine if they could in fact generate the original sentence. If so, that set of basic trees constituted a correct analysis of the sentence. If not, the set of basic trees was discarded and another set tried.

Apparently the reason for stages 3 and 4 was that the analysis in stages 1 and 2 was not refined enough. Furthermore, the MITRE parser, like the other two discussed, has serious problems with spurious ambiguity; for example, one 12-word sentence had 48 presumed surface trees after stage 1.

One can envision a natural langage analysis system consisting of modules for surface, reverse transformational, and semantic analysis. These modules could perform their work in turn or their functions could be intermixed. In any case, the lessons of the three projects described are that the surface grammar must be extremely detailed and that a purely context-free model is not appropriate for its expression.

The parsing system of the Linguistic String Project at New York University under Naomi Sager grew out of the earlier UNIVAC I parsing program at the University of Pennsylvania,

developed in conjunction with Zellig Harris' string analysis.[11] Just as string analysis is intermediate between immediate constituent analysis and transformational analysis, so the grammar developed by Sager based on strings can be analyzed by the methods for a context-free grammar, but in addition it makes explicit the information in a sentence needed to perform transformational analysis. String analysis thus provides solid linguistic grounds for a surface grammar.

The Linguistic String Project grammar[12] has overcome the difficulties of a large unwieldy purely context-free grammar by having instead a smaller context-free grammar which indicates the permissible constructions only in broad outline. There are only on the order of two hundred productions and no more than twenty terminal symbols, indicating the "parts of speech," but each word in the lexicon may have various attributes assigned to it as well. The context-free component is augmented by another component consisting of restrictions to handle the finer detail. These restrictions check aspects of the tree structure so far evolved together with attributes of core words in the sentence, and on this basis permit or disqualify but never modify constructions produced by the context-free component. As proved in Section 5.11, a grammar thus defined could be converted into a purely context-free grammar which would recognize the same language. However, this context-free grammar would be several

[11] Zellig Harris, String Analysis of Sentence Structure, The Hague, 1962, pp. 58-65.

[12] Naomi Sager, "A Two-Stage BNF Specification of Natural Language," Journal of Cybernetics, Vol. 2, 1972, pp. 39-50.

orders of magnitude larger, much larger in fact than the Kuno and Oettinger and Robinson grammars.

The Sager grammar is explained in more detail in Chapter 2.

6.3. Parsing Algorithms.

Following the specification of ALGOL in Backus-Naur form (BNF) in the ALGOL 60 report,[13] an interest arose in syntax-directed compiling. In syntax-directed compiling, the syntax of the language is specified independently of the code in the compiler and treated as parametric data for the compiler. Coincident with this development was a growth of interest in algorithms for parsing a language according to a syntax-specification or grammar.

Two classes of parsing techniques appeared almost immediately -- top-down and bottom-up. Early versions of the top-down strategy appeared in articles by Floyd[14] and by Cheatham and Sattley.[15] Briefly, a top-down parser proceeds from left to right generating a tree from the root symbol of the grammar and seeking to match it with the input string. When no match is possible the parser backs up and

---

[13] Peter Naur (ed.), "Report on the Algorithmic Language ALGOL 60," Communications of the ACM, Vol. 3, 1960, p. 299.

[14] Robert W. Floyd, "The Syntax of Programming Languages -- A Survey," IEEE Transactions on Electronic Computers, Vol. EC-13, pp. 346-353, August 1964.

[15] T. E. Cheatham and K. Sattley, "Syntax-Directed Compiling," Proceedings -- Eastern Joint Computer Conference, AFIPS, Vol. 25, pp. 31-57, 1964.

attempts to generate another tree. If a tree is matched with the entire input string, a valid parse has been found. If all paths in the grammar have been exhausted without matching the entire input string, no parse is possible.

A bottom-up parsing algorithm was used in the first account of syntax-directed compiling by E. T. Irons.[16] Bottom-up parsing proceeds from left to right, grouping symbols in the input string. If successive symbols spanning contiguous portions of the input string constitute the right side of a production in the grammar, the symbols may be coalesced and replaced by the symbol on the left side of that production. If no further coalescing is possible, the parser backs up and tries alternative ways of coalescing symbols. If the root symbol is produced in this manner, a valid parse has been found. If all possible paths in the grammar are exhausted without producing the root symbol, no parse is possible.

Floyd[17] developed a fast bottom-up parsing technique for most programming languages based on the precedence relations used implicitly by early compilers. By defining precedence relations on the operators in a language (e.g. "+", "*", "/") one could isolate the most deeply nested portions (called the "handle") of the input string and coalesce them first.

---

[16] E. T. Irons, "A Syntax-Directed Compiler for ALGOL 60," Communications of the ACM, Vol. 4, pp. 51-55, Jan. 1961.

[17] Robert W. Floyd, "Syntactic Analysis and Operator Precedence," Journal of the ACM, Vol. 10, pp. 316-33, July 1963.

Studies[18] have indicated that this parsing technique is the best one known for most programming languages. Unfortunately, it does not apply to ambiguous grammars or grammars which have no clearly defined precedence relationships, and therefore could not be applied to natural languages.

One difficulty with the top-down and bottom-up parsing techniques described above is that theoretically they may require exponential time, since all possible paths in the grammar may have to be searched. In practice, with programming languages this can usually be avoided by judicious ordering of the alternatives and by application of simple heuristics at crucial points in the parsing process. However, when one of the goals of the analysis is to produce all possible parses of an input string, as it frequently is in natural language processing, the time required is necessarily exponential.

An algorithm known as the nodal spans parsing algorithm was developed by John Cocke[19] for use in the RAND parsing program PARSE. This algorithm is bottom-up, can be used with any grammar, and produces all parses in time n^3. Briefly, this savings is achieved by producing all parses

---

[18] See for example Bruce Knobe, An Experimental Comparison of the Efficiency of Parsing Algorithms, Ph.D. Thesis, New York University, June 1972.

[19] Described briefly in David G. Hays, "Automatic Language-Data Processing," in Computer Applications in the Behavioral Sciences, Harold Borko, ed. Englewood Cliffs, N. J., 1962, pp. 411-414.

in parallel. If a portion of the input string is capable of two different interpretations which in turn can be subsumed under the same symbol, this symbol is represented by only one node in the tree-like structure built up by the parser. Ambiguity is thus localized and prevented from causing exponential growth. A detailed account of the algorithm is given in Chapter 3.

A drawback of nodal spans parsing is that it requires n^3 space as opposed to linear space for top-down parsing. Of course, if the space is available this should not be a concern. However, it will be seen dramatically in Chapter 4 that space considerations do indeed become crucial. On the other hand, if difficulties with space can be controlled, the gain in time from exponential to n^3 would certainly justify the efforts.

Another advantage of a nodel spans parser would be its convenient representation of ambiguity.

1.4. Ambiguity in Natural Language

Probably the majority of English sentences are ambiguous, at least within the level of grammatical refinement possible today. The meaning or reference of a word or phrase or the structure of a phrase may be uncertain in a smaller context and only become certain in a larger context. For example, the word

"... types..."

can either be a plural noun or a singular verb. Only when we

look at larger context is this ambiguity resolved:

"he types"

"the types"

There seems to be no limit on the amount of context necessary to resolve ambiguities of meaning, structure, or reference. Knowledge of idioms and selectional restrictions suffice to resolve the structural ambiguities of

"I avoid wiating rooms."

"I avoid waiting tables."

The structurally ambiguous phrase

"The experiment of Michelson and Einstein on special relativity ..."

is disambiguated by syntactic analysis of the rest of the sentence:

"The experiment of Michelson and Einstein on special relativity are cornerstones of modern physics."

Sometimes sublanguage data is required, for example in the disambiguation of the final sentence in a surgery report:

"The patient left the operating room in good condition."[20]

(Compare: "The janitor left the operating room in good condition.") One might be able to resolve the structural ambiguity of

"I don't like visiting relatives."

by considering somewhat more context, e.g.

---

[20] I. D. J. Bross, P. A. Shapiro, and B. B. Anderson, "How Information is Carried in Scientific Sub-Languages," Science, June 1972, pp. 1303-7.

"Unfortunately, Uncle Yorick came to visit us this weekend. I don't like visiting relatives."

"Unfortunately, we have to go to Philadelphia to visit Uncle Yorick. I don't like visiting relatives."

but this would require such detailed linguistic information as the partial synonomy relations of "uncle" - "relative" and "I" - "we" and assumptions on inter-sentence relationships, perhaps imposed by the adverb "unfortunately". Rather detailed knowledge of current conditions in America, which one would not ordinarily expect to find in the given text would be needed to disambiguate the sentences

"The elderly people who walked through the park at night needed protection from the police."

"The arrested demonstrators needed protection from the police."

The ambiguities of reference of the sentences

"I hid the flashlight in the kitchen under the sink."

"I saw the man in the park with a telescope."

could not be resolved without geographical knowledge so localized as to be virtually unavailable. And of course, some sentences will be so inherently ambiguous whether intentionally or unintentionally that no amount of context and no degree of detail in analysis will serve to disambiguate them.

In any natural language processing system, therefore, the decision must be made whether to have the syntactic

component output all possible parses or merely the most likely parse. The latter position has been argued by analogy with assumed human thought processes, but while such analogies would seem a fruitful source of inspiration in computational linguistics, they do not appear particularly relevant as arguments for the shape of a specific system which is not meant as a cognitive model.

In some situations it is acceptable to aim for the one intended parse and neglect the rest, for example in on-line question-answering systems in which the answer is couched in terms which indicate the parse selected. This might be accomplished, with varying degrees of success, by means of careful ordering of alternatives in a context-free grammar, examination of certain amounts of context, an appeal to statistical probabilities, or an exit to a routine which asks the user for help. In fairly restricted systems a semantic component can carry much of the burden of disambiguation; however, in systems with a broader scope, for example the systems which would be necessary for effective information retrieval from scientific literature, such a detailed semantic component would become impossibly unwieldy. Work has been done on disambiguation by means of selectional restrictions which hold true for specific sublanguages, e.g. the sublanguage of theoretical pharmacology[21] and the

---

[21] Naomi Sager, "Syntactic Formatting of Scientific Information," AFIPS Conference Proceedings, Vol. 41, 1972, p. 791.

sublanguage of surgery reports.[22] This approach has proved useful, but it is unlikely it can be extended very successfully to less restricted areas in which language is used with less precision. Complete disambiguation of sentences from reasonably general texts, therefore, seems beyond our grasp at this point.

If we wish our syntactic component then to output all possible parses, we would like this information presented in a lucid manner.

Consider for example the sentence

"I believe he hid the flashlight in the kitchen under the sink." (3)

It is not clear whether he took the flashlight from the living room and hid it in the kitchen under the sink, or took the flashlight that's usually kept in a cupboard in the kitchen and hid it under the sink, or took the flashlight which is usually kept in the kitchen under the sink and hid it in the bathroom. In fact, in the absence of any semantic information, it could be that the kitchen is under the sink. The sentance thus has five parses, but to say this is to disguise the fact that its ambiguity is localized. It is only the reference of the prepositional phrases that is uncertain. A top-down parser lists each parse separately, and one can see how they differ only through node-by-node comparison. The nodal spans algorithm, by contrast, gives

[22] I. D. J. Bross, et al., Op. Cit.

all parses simultaneously and in a concise from that pinpoints the source of the ambiguity. The abbreviated representation of the parses of sentence (3) shown below can be read directly off the tree structure generated by the nodal spans algorithm.

1.	<SUBJECT>	<VERB>	<OBJECT>	
	I	believe	2.	
2.	<SUBJECT>	<VERB>	<OBJECT>	<RV>[23]
A.	He	hid	the flashlight	3A. 4.
B.	He	hid	the flashlight 3A.	4.
C.	He	hid	the flashlight 3A. 4.	
D.	He	hid	the flashlight	3B.
E.	He	hid	the flashlight 3B.	
3.	<*P>	<NSTGO>		
A.	in	the kitchen		
B.	in	the kitchen 4.		
4.	<*P>	<NSTGO>		
	under	the sink		

The nodal spans algorithm also permits finer classification of words. Suppose for example that a word in the sentence being parsed has six different meanings and the following word has five meanings. A top-down parser looking

[23] The notation here is adopted from the Sager grammar. <RV> stands for right adjunct of the verb, <*P> for a preposition, and <NSTGO> for a noun string in the object position.

for all parses needs thirty passes over these two words, and of course for longer sentences the number of passes grows exponentially. For this reason there is very little breaking of words into their various meanings with a top-down parser. In a bottom-up parallel parser, nodes for all the meanings can be generated in one pass, eliminating the need for backtracking and permitting greater refinement of word meaning.

In a system whose only goal is parsing, a word can be broken into as many different meanings as have different sets of attributes. In a mechanical translation system, the words can be broken into as many different meanings as have different translations in the target language. This can help resolve the problems of multiple meanings in mechanical translation, for if only two of six possible meanings of a word can participate in a final parse of the sentence, we have significantly restricted the number of translations we need consider for that word.

The parser in current use in the Linguistic String Project is a naive top-down back-up parser. Generally the first parse is found rather rapidly and in a surprisingly high proportion of the cases this is the preferred parse. On the other hand, to find all possible parses requires exponential time, which means in practice that for long sentences, all parses are not found. These considerations motivate the development of a nodal spans parser for the Sager grammar.

One would suspect that since a great deal of the power in the Sager grammar resides in the restrictions, that the restrictions must be expressed in such a way that they can work in a nodal spans environment. The statistical results of Chapter 4 on nodal spans parsing without restrictions show that this is indeed the case. Some suggestions for a metalanguage for expressing grammatical constraints were outlined by J. T. Schwartz,[24] who also indicated ways in which the nodal spans algorithm could be modified to handle restrictions expressed in such a language.[25] In Chapter 5 of this work such a metalanguage is described and defined. In Chapter 6 and in Appendix B, the application of this metalanguage to the restriction component of the Sager grammar is described.

The nodal spans algorithm also lends itself nicely to an algorithmic treatment of semigrammaticality. This is described in Chapter 7.

---

[24] J. T. Schwartz, "The Analysis of Natural Language," unpublished manuscript, undated.

[25] J. T. Schwartz, "Generalized Nodal Span Parse Routine: Preliminary Draft," SETL Newsletter # 46, 1971.

CHAPTER 2. THE GRAMMAR

2.1. The components of the Grammar

The grammar used by the Linguistic String Project at New York University under Naomi Sager was developed by Sager in 1962-63 as a modification of the University of Pennsylvania UNIVAC grammar to get all possible parses of a sentence. The system was first programmed in 1964-65 and has since gone through several reprogrammings. The system and the grammar have been used in a number of studies, including investigations of information extraction from physics texts, foreign language teaching, and the sublanguage of pharmacology.

The grammar consists of four components:

1. A context-free grammar, written in Backus-Naur Form (BNF). It consists of about 200 productions with about 200 nonterminal symbols and about 20 terminal symbols representing categories or "parts of speech". As stated above, this component specifies in broad outline the syntactic constructions which occur in English.

2. Restrictions, written in a special-purpose metalanguage. There are about 200 restrictions. They supply the necessary detail which the BNF component omits. This is perhaps the most powerful component in the grammar, for eliminating it would require an expansion of the BNF component by several orders of magnitude.

3. Locating relations. In the top-down parser, the restrictions find their arguments by means of about 30 locating relations which operate on the nodes of the tree. These relations are defined either in a special-purpose metalanguage or as FORTRAN subroutines.

4. The word dictionary, consisting of about 10,000 entries including inflected forms. Individual words are listed as tree structures whose nodes are the various categories and attributes that a word can have.

Each of these components will be described in turn. The BNF component will be described in somewhat greater detail than the other components, for we will use it as an opportunity to take a longer look at the English language and some of the issues that come up in its syntactic analysis to see how they have been resolved in this formal grammar. No attempt is made to list all the constructions in the grammar, but only enough to impart its scope and flavor.

3.2. The BNF Component [1]

1. The Assertion: The English sentence consists of a center string followed by an endmark (".", ";", "?", "!"), and possibly preceded by an introducer ("And", "But", "Yet", etc.). The most important center string is the assertion. The assertion consists of a subject followed by a verb

---

[1] This section is, for the most part, a summary of Naomi Sager, String Program Report No. 4: A Computer String Grammar of English, Parts I and II, New York, 1968.

followed by an object. In addition, the sentence adjuncts may be inserted at the beginning or the end of a sentence or between, and in some cases even within, the principal constituents of the sentence. For instance, in the sentence

"For instance, in the sentence ... is an initial sentence adjunct."

"For instance" is an initial sentence adjunct. Furthermore, the verb may have a right adjunct to the right of the object, as the word "indefinitely" in the sentence

"I can continue this indefinitely."

Since this right adjunct is not adjacent to the verb it must appear in the definition of the assertion at the same level as the principal constituents and the sentence adjuncts.

The definition of the assertion in the grammar is therefore

<ASSERTION> → [<SA>]<SUBJECT>[<SA>]<VERB>[<SA>]<OBJECT>[<RV>][<SA>].[2]

Here <SA> stands for Sentence Adjunct, <RV> stands for the Right adjunct of the Verb, and the brackets enclosing an element indicate that it is optional.

This analysis of the assertion differs from the usual phrase structure analysis, in that it places subject, verb, and object all on the same level of the tree. In the usual phrase structure analysis[3] the verb and object are at a

---

[2] An account of the mnemonics used in the grammar is given at the beginning of Appendix A.

[3] See for example Noam Chomsky, Syntactic Structures, The Hague, 1957, p. 26; or H. A. Gleason, Linguistics and English Grammar, New York, 1965, p. 141.

deeper level than the subject, appearing as constituents of the "verb phrase":

$$S \rightarrow Np_1 \quad Vp$$

$$Vp \rightarrow Verb \quad Np_2$$

There would appear at first sight to be no compelling reasons for preferring one analysis over the other. However, it will be shown in Chapter 6 how the requirements of the nodal spans algorithm force a modification of the string grammar definition in the direction of the phrase structure analysis.

2. The Verb: Given the sequence of verbs and verb-like words in the sentence

"They may have simply been trying to appear to be pressing for action." (1)

the question immediately arises as to what should be called the main verb. The possibilities include

1) Only the first verb-like word (in this case, "may").

2) The first inflected verb (but not a form of "do" preceding an infinitive form) or the first verb following a modal such as "can", "may", etc. (here the verb "have" following the modal "may").

3) The first verb or verbal following a legal sequence of forms of the auxiliary verbs "have", "be", and "do" (in this case, "trying").

4) The last verb or verbal in the sequence (in this case, "pressing").

The question is merely one of choosing the solution most in line with the goals and simplicity of the system.

In favor of solution 1) is the fact that number-agreement restrictions with the subject act only on the first verb or verb-like word; if the first verb-like word is uninflected, like "may", then no number agreement is required for any word in the verb sequence. In favor of solution 4) is that subject - verb and verb - object selectional restrictions act on the final verbal element, and the related reason that the above sentence is transformationally derived from the sentence

"They press for action."

Perhaps solution 3) corresponds best to the intuitive notion elementary education has given us of what the main verb should be.

In this grammar, solution 2) is chosen. The modal words ("will", "can", "may", etc.) seem insufficiently verb-like to be called the main verb. They take no affixes (* "He mays ...") and they cannot apepar alone unless a following verb is understood ("I shall."). Therefore, they are considered preverbal tense words. As to solution 3), it is true that the verbs "have" and "be" may be either auxiliary verbs or principal verbs (a fact which makes the sentence "They are visiting relatives." ambiguous). However, the verbal objects of "have" and "be" generally occur in the same environments as noun objects:

"He has arrived."

"He has money."

"This man is causing me trouble."

"This man is my uncle."

and it is simpler computationally to consider "have" and "be" as verbs in all their occurrences, rather than verbs sometimes and sometimes auxiliaries. The main reason for choosing solution 2) over solution 4) is that it allows greater regularity in the statement of the grammar. For example, the gerund string <VINGO> is defined as a present participle <*VING> [4] followed by an object. This string can then be used in the analyses of all the following sentences:

"Visiting relatives is an activity I enjoy."

"I enjoy visiting relatives."

"I am visiting relatives."

Under solution 4) in the last two sentences "visiting" would be classed as part of the verb and "relatives" as part of the object. However, the following concession is made to solution 4): In the top-down parser the last verb in the sequence is defined as the "deepest verb" and in the bottom-up parser it is brought up as the core of the verbal object for use in selectional restrictions.

In line with solution 2) the verb is defined as

<VERB> → <LTVR> / <LWR>[<SA>]<LVR> .

---

[4] The "*" indicates a terminal or category symbol or a "part of speech".

(Throughout the grammar <LXR> stands for the element X surrounded by its Left and Right adjuncts. Thus <LTVR> is a Tensed Verb flanked by any left and right adjuncts. <LWR> is a tense word W similarly flanked and <LVR> the same for an uninflected verb V.) The structure assigned by the grammar to sentence (1) is given in Fig. 1. (Null adjuncts have been removed.)

In the graph of Figure 1, as throughout this work, parse trees are represented in binary form. This reflects the internal organization of the grammar and the parser, and also emphasizes the importance of the coelement relation.

In the graph, <LVENR> represents the past participle VEN with its adjuncts, <VENO> the past participle followed by an object, <LVINGR> represents the present participle, <TOVO> the infinitive construction "to"-Verb-Object, and <PN> the prepositional phrase Preposition-Noun.

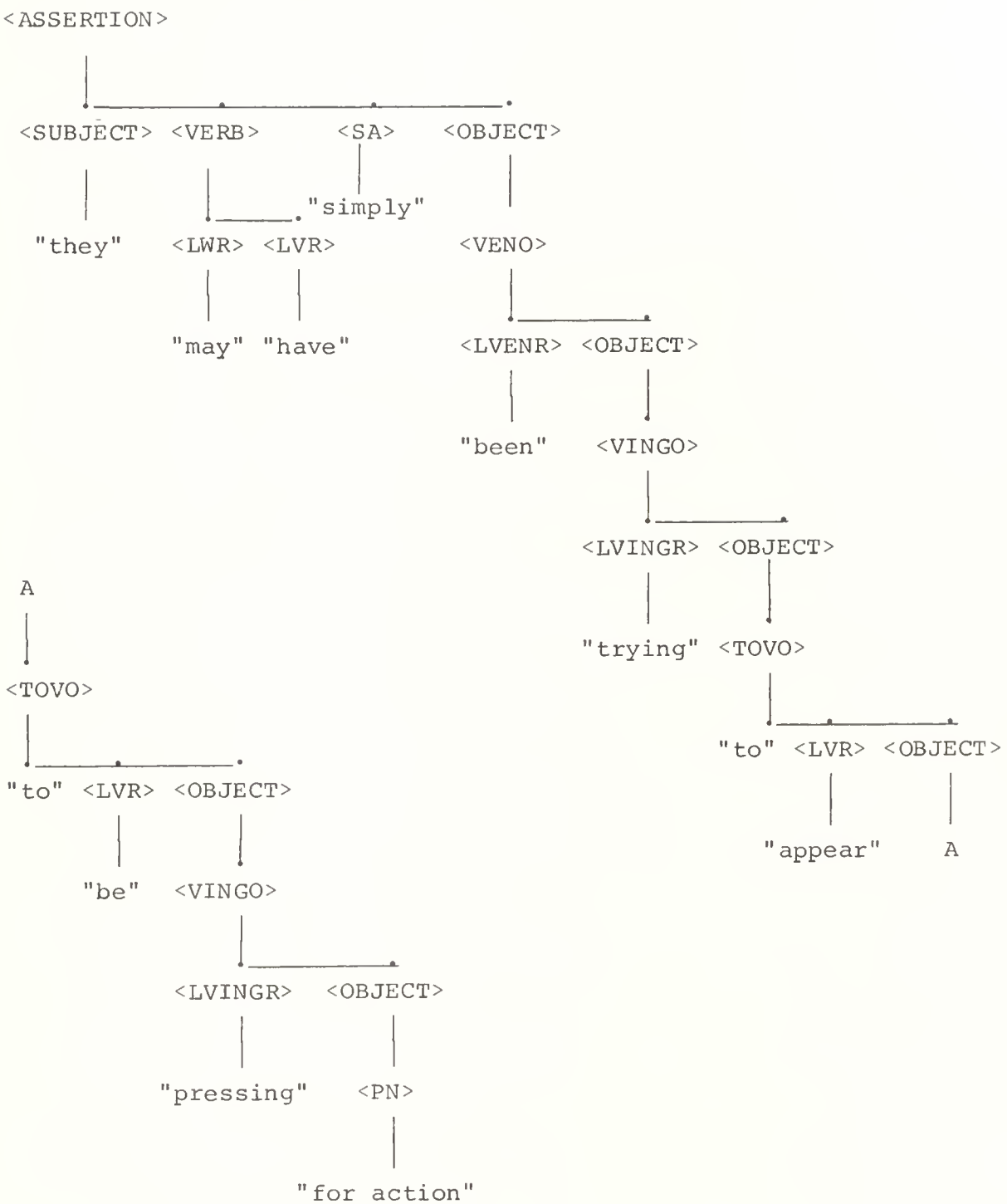
3. The Subject and Object: Noun Phrase and Nominalizations: The subject and object positions of the assertion and other strings can be filled by simple noun phrases or by more complex verbal and sentence nominalizations. Among the simple noun phrases are nouns with left and right adjuncts

"the mysterious lady in the black cape", (<LNR>) [5]

pronouns with adjuncts

[5] The notation used in the machine grammar for a construction will appear in parentheses after the example of it.

Figure 1


<ASSERTION>
<SUBJECT>
<VERB>
<SA>
<OBJECT>
"they"
<LWR>
<LVR>
"simply"
"may"
"have"
<VENO>
<LVENR>
<OBJECT>
"been"
<VINGO>
<LVINGR>
<OBJECT>
"trying"
<TOVO>
"to"
<LVR>
<OBJECT>
"appear"
A
A
<TOVO>
"to"
<LVR>
<OBJECT>
"be"
<VINGO>
<LVINGR>
<OBJECT>
"pressing"
<PN>
"for action"

"only you" (<LPROR>)

and names

"George Bernard Shaw" (<NAMESTG>)

In place of the noun, nominalizations of verbs may occur, both st ong nominalizations

"a pleasant walk"

"utter confusion"

for which there seem to be no regular derivation rules and now appear in the dictionary as nouns; and weak nominalizations, using the gerund form

"whiskey drinking" (<LVINGR>)

Less noun-like sentence nominalizations can also occur:

"The drinking of Irish whiskey can be baneful to one's health." (<VINGOFN>)

"Your drinking Irish whiskey can be baneful to your health." (<NSVINGO>)

These constructions cannot satisfactorily be explained as gerunds filling the noun's place in a noun phrase, for if the verb is necessarily transitive, the "of" phrase cannot be deleted, as noted before in the discussion of excision.

The most common among weak sentence nominalizations, especially in the object position, is the word "that" followed by an assertion.

"That Connally will run in 1976 is inevitable." (<THATS>)

"I know that Connally will run in 1976." (<THATS>)

In the object position "that" can be omitted. A similar

construction requires a subjunctive verb or "should" in the assertion and occurs in different environments, i.e. with different predicates:

"That Kennedy run in 1976
is imperative." (<ClSHOULD>)

Another sentence nominalizing construction is accomplished with "for -- to":

"For Kennedy to have run against Nixon in 1972
would have been foolish." (<FORTOVO>)

The <TOVO> construction, at least in the subject position, may be thought of as the <FORTOVO> construction with an omitted "for one":

"(For one) to have run against Nixon in 1972
would have been foolish." (<TOVO>)

An important class of sentence nominalizations is the class of wh-strings. In all of them, an assertion is preceded by a group of words including a wh-word indicating part or all of a noun phrase in the assertion, the noun phrase then being "omitted" or "zeroed" from the assertion. A noun may be omitted from the subject or the object of the assertion or the object of a preposition:

"Who ( ) stole it is uncertain." (<WHS-N>)

"What he stole ( ) is uncertain." (<WHS-N>)

A prepositional phrase may be omitted from an object or from certain adjuncts, and the assertion preceded by the preposition and a wh-word:

"On whom we can rely ( ) is a good question."
(<PWHS-PN>)

"With what we should cut the samami
is a good question." (<PWHS>)

Instead of the entire noun phrase being zeroed, only its qualifier may be zeroed. A wh-word followed by the noun precedes the assertion. The above four possibilities occur in this case as well:

"Which man ( ) stole it is uncertain." (<WHNS-N>)

"Whose watch he stole () is uncertain."(<WHNS-N>)

"On which candidate we can rely ( ) is
a good question," (<PWHNS-PN>)

"With which knife we should cut the salami
is a good question." (<PWHNS>)

It is possible for an indication of affirmation or negation, rather than a noun phrase, to be zeroed. Frequently, "or not" occurs in these constructions:

"Whether I will vote this time is still up
in the air." (<WHETHS>)

"Whether or not to vote for the lesser of two evils
is a moot question." (<WHETHTOVO>)

There is another class of wh-strings which can occur in the subject and object positions which are to be distinguished from the above sentence nominalizations. These are the noun-equivalent wh-strings, which occur in the same environments as nouns:

"What ( ) is lost is lost forever." (<WHATS-N>)

"I just came from where you are going." (<WHERES>)

"Do whatever you do ( ) with confidence." (<WHEVERS-N>)

Some occurrences of these clauses are open to ambiguity since they may be used sententially.

"What ( ) was lost in the war can never be determined." (<WHS-N>)

or as noun equivalents,

"What ( ) was lost in the war can never be regained." (<WHATS-N>)

The sentence

"What he writes is unclear."

is capable of two interpretations, the sentential, "It is unclear what sort of things he writes." and the noun-equivalent, "Everything I've read of his has been unclear."

There are other sentence nominalizations which occur only in the object position, usually taking an accusative subject:

"She asked him to get out of her life." (<NTOVO>)

"She made him leave." (<SVO>)

Important among these are the sentence nominalizations having an understood verb "to be":

"I consider that the last straw." (<SOBJBE>)

"I regard that as the last straw." (<SASOBJBE>)

Most of these strings occur only in very restricted environments.

4. Other Object Strings: The object carries much of the weight of the grammar. We have already seen how much of a complex verb sequence is shunted into the object position.

Another example is given by the alternative <*NULLOBJ> for <OBJECT>, which is simply a null terminal symbol. There are in English two kinds of assertion -- the transitive and the intransitive. In the machine grammar the intransitive assertion is realized by taking the <*NULLOBJ> alternative for <OBJECT>. It would perhaps be more principled to specify two separate kinds of assertion instead, but using <*NULLOBJ> seems more economical, for as well as having two assertions, we would also need two <VENO>'s, two <VINGO>'s, and two of every other string in which <OBJECT> appears.

The indirect object is combined with the direct object and listed as the following alternative of <OBJECT>:

<NN>→ <NSTGO>[<SA>]<NSTGO> .

"He gave the alumni fund a large contribution."

Here <NSTGO> represents a Noun String appearing in an Object position.

Prepositional phrases usually occur in adjunct positions but some verbs take prepositional phrases as objects:

"Universities today are in deep financial trouble." (<PN>)

"They cannot rely on individual contributors." (<PN>)

Other subclasses of verbs take more elaborate constructions involving prepositions, for example, a noun followed by a prepositional phrase,

"You cannot attribute their difficulties to mismanagement." (<NPN>)

and a noun followed by a preposition and nominalized sentence,

"Don't ask them about their investing in South African diamonds." (<NPSVINGO>)

An adverbial preposition is a preposition which changes the meaning of the verb to which it is appended, e.g. "look up", "find out", "mix up", "rip off", "carry on". Perhaps the most intuitively satisfying way of handling these would be to consider them part of the verb. However since other elements in the sentence can intervene between a verb and its adverbial preposition, as in

"Please look the number up."

it is more convenient for computational purposes to consider the adverbial preposition part of the object. The adverbial preposition constructions have a variety comparable to the prepositional constructions, from the simple word or phrase,

"Give up." (<DP1>)

"Look up the number." (<DP2>)

to the more complex noun - adverbial preposition - prepositional phrase,

"He always mixes my name up with the name of his only other friend." (<DP3PN>)

Altogether there are nearly sixty alternatives for <OBJECT>. Stored with each verb in the dictionary is a list of the alternatives it may take.

5. Sentence Adjuncts: Sentence adjuncts modify the entire sentence or relate it to other sentences in the text. There are four categories of sentence adjuncts. The first, the short sentence adjuncts, includes adverbs,

"He generally skis well." (<DSTG>)

prepositional phrases involving nouns or adjectives,

"In theory at least he knows how to ski." (<PN>,<PA>)

time nouns,

"He skied every day last winter." (<NSTGT>)

and a group of adverbs which because they cannot be modified by other adverbs are classed as interjections in the grammar,

"Moreover, he won three medals." (<*INT>)

The other three categories are the longer strings which can be attached only to the center-like strings. There is the important group of subordinate clauses. A subordinate conjunction can be followed by an object of the implied verb "to be",

"Though blind, he was a top-notch private eye." (<SUBO>)

an assertion,

"Because he had intimate knowledge of the underworld, he solved every case quickly." (<SUB1>)

a past participle indicating passive,

"Although captured by the syndicate, his courage did not fail him." (<SUB2>)

a present participial construction,

"Before dropping his gun, he was prepared to fight it out." (<SUB3>)

and a subject - object-of-be construction with an implied "to be",

"With his gun out of reach, his situation appeared hopeless." (<SUB6>)

Closely related to the last of these is the third category, subordinate clauses with no subordinate conjunction heading it:

"The capital destroyed, a surrender was considered forthcoming." (<SOBJBESA>)

"A career military man, the new president was sworn in in uniform." (<OBJBESA>)

There are a few wh-strings which can appear as sentence adjuncts:

"Whatever you say (), I'll do as I please." (<WHEVERS-N>)

"She quit smoking, which surprised us." (<SAWHICHSTG>)

For the most part sentence adjuncts may be adjoined at any point in the sentence:

"In general, his lectures are good."

"His lectures, in general, are good."

"His lectures are good, in general."

There are exceptions however. The clause "which surprised us", for example, cannot appear at the beginning of a sentence.

6. Left and Right Adjuncts: Except for the left and right adjuncts of nouns, most left adjuncts are adverbs,

"rarely drinks" (<DSTG>)

"hardly reliable" (<DSTG>)

and most right adjuncts are adverbs or prepositional phrases,

"reliably enough" (<DSTG>)

"drinks in large quantities" (<PN>)

The structure of the left adjunct of the noun is indicated by the following production:

<LN> → <TPOS> <QPOS> <APOS> <NSPOS> <NPOS>

as in

"The four standard children's test programs."

Here <TPOS> represents the position for the article, but may also be occupied by a possessive ("John's"), an interrogative or relative pronoun (whose, which), or the string "how many of the". Furthermore, an article or possessive can be preceded by a quantifier (as a left adjunct) as in "all the", "many a", "only John's". <QPOS> is the quantifier posisition, as in

"the three new sailboats".

<APOS> is the position for a string of adjectives, and may also be occupied by a present participle

"the flying Dutchman",

by a past participle,

"a fallen angel",

or by certain noun phrases indicating measure,

"a size ten dress".

<NSPOS> is the post-adjectival position for the possessive noun. In this position the possessive generally indicates purpose rather than possession. The distinction between

the <TPOS> and the <NSPOS> possessive can be seen in the phrase

"my little daughter's grown man's shirt".

<NPOS> is the position for a noun acting as an adjective, as in

"feast day"

or

"rum running".

The validity of this analysis of the left adjunct of the noun can be seen by trying to permute the elements, e.g.

* "three the new sailboats"

* "the new three sailboats" (except in very special contexts)

* "a feast happy day".

As with <ASSERTION> the analysis of <LN> differs from the usual phrase structure analysis of the noun with its modifiers, in that it places all the elements on the same level. Phrase structure analysis, by contrast, analyzes the noun phrase as shown in the figure below.[6]

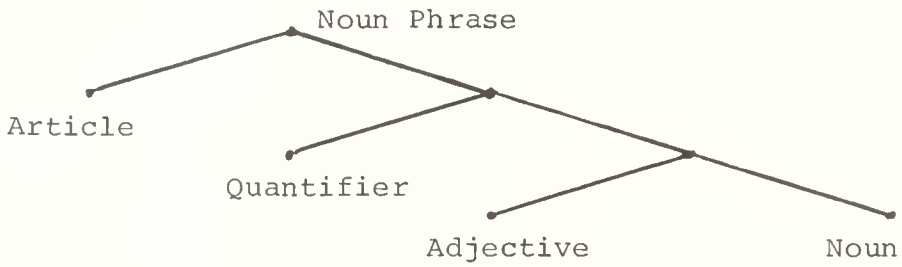


Figure 2

[6] H. A. Gleason, Jr., Linguistics and English Grammar, New York, 1965, pp. 151-9.

In Chapter 6 it will be shown how the requirements of the nodal spans algorithm force a refinement in the string analysis of the noun phrase but not quite in the manner of traditional phrase structure analysis.

Constructions which can appear in the right adjunct of the noun are generally the same as can appear as the object of the verb "to be" but often in somewhat more restricted environments. Thus, corresponding to the noun phrase object-of-be,

"Picasso was a famous painter."

is the appositive right adjunct,

"Picasso, the famous painter," (<APPOS>)

But appositives are rare after most pronouns:

∃? "He, the famous painter"

Similarly, adjectives can appear as objects-of-be,

"The people are free."

but only a few can appear alone as right adjuncts of nouns while others have to be modified in some way:

∄ "The people free"

"The people responsible" (<ADJINRN>)

"The people, free at last from the yoke of oppression." (<ADJINRN>)

Prepositional phrases may occur in much less restricted environments:

"Universities are in deep financial trouble."

"Universities in deep financial trouble need government aid." (<PN>)

Various verbal phrases may occur in the right adjunct of nouns, including the past participle indicating passive,

"The man sent a letter sent a letter." (<VENPASS>)

and present participial constructions,

"The man hitting the sidewalk is my uncle." (<VINGO>)

Assertions omitting a noun and possibly preceded by "that" can occur:

"The man that () came to dinner." (<THATS-N>)

"The man we gave the dinner for ()." (<S-N>)

In addition many of the wh-strings which are sentence nominalizations can also occur as right adjuncts of nouns, and here in a considerably wider variety of contexts since the nouns they refer to need not be sentential nouns:

"The bird which () has red tailfeathers." (<WHS-N>)

"The tree in which it has built its nest." (<PWHS>)

"The frog whose croaking we hear ()" (<WHNS-N>)

Some of the constructions which appear as right adjuncts of nouns also occur as right adjuncts of adjectives.

7. Other Center Strings: Questions: There are two sorts of questions -- those asking for a choice or an affirmation or negative, and yes-no questions, and those asking for a specific part of an assertion to be filled in, the wh-questions. The yes-no question is formed by splitting the verb into two parts, the first normally a tense word or a tensed form of "do", "be", or "have", preceding the subject, and the second either a tenseless verb or null following the subject:

"Will I ever understand you?" (<YESNOQ>)

"Was it all true?" (<YESNOQ>)

The yes-no question corresponds to the sentential nominalization containing "whether".

A wh-question consists of a wh-word, possibly preceded by a preoposition and possibly followed by a noun, followed by a yes-no question or an assertion from which a noun phrase or prepositional phrase has been omitted. The possibilities that occur correspond exactly to wh-sentence-nominalizations. A noun phrase may be omitted from the subject or the object:

"Who () stole it?" (<WHQ-N>)

"Which man () stole it?" (<WHNQ-N>)

"What did he steal ()?" (<WHQ-N>)

"Whose watch did he steal ()?" (<WHNQ-N>)

A prepositional phrase may be deleted from the object or an adjunct:

"On whom can we rely ()?" (<PWHQ-PN>)

"On which candidate can we rely ()?" (<PWHNQ-PN>)

"With what should we cut this salami?" (<PWHQ>)

"With which knife should we cut this salami?" (<PWHNQ>)

Permutations: Almost any permutation of the normal ordering of an English sentence can occur, especially in poetic writing. As Chomsky has pointed out,[7] these are more likely stylistic inversions rather than separate grammatical construc-

[7] Chomsky, Noam, Aspects of the Theory of Syntax, Cambridge, Mass., 1965, pp. 227-228.

tions. In the machine grammar only two of the most common permtuations have been implemented.

In the first, a tense word or a tensed auxiliary follows "neither", "nor", an adverb, or a prepositional phrase, and is followed by the subject. The emphasis is often negative in such sentences:

"Little did I suspect she was a Russian spy." (<TSUBJVO>)

"Nor did I see her very often."

"In no case will I tell you more."

The second permutation is of the form "Object-of-be is Subject":

"In London are several fine museums." (<OBES>)

"Finer still are the museums of Paris."

"Burnt to a crisp was the bacon she gave me."

Imperative: The imperative construction is characterized by a zeroed subject and an uninflected main verb. It has not yet been implemented in the machine grammar since it occurs rarely in the scientific literature the parsing program has been used for so far.

The complete BNF component together with an explanation of the mnemonics, examples of the strings, and an index of the non-terminal symbols is given in Appendix A.

2.3. The Restrictions.

For the top-down parser the restrictions are written in a metalanguage which is a subset of English (slightly augmented) so they can be read easily by human beings and

can be compiled by the machine into stacks of FORTRAN routines which execute the locating relations. The restrictions are called at various points in the construction of the tree either to check parts of the tree already built and upon failure to tear up that part (well-formedness restrictions) or else to check that certain conditions are satisfied before attaching an alternative as a node in the first place (disqualifying restrictions).

For the bottom-up parser the restrictions are written in the metalanguage described in Chapter 5. In this formulation there is no checking of the tree beyond the immediate nodes; rather flags which are set, passed up the tree, and checked during parsing, carry all the relevant information.

The restrictions handle the context-sensitive aspects of natural language, generally by finding grammatically related words and checking that they have compatible word subclasses.

One can identify several types of restrictions according to the role they play in the grammar. The complete set of restrictions together with comments is given in the nodal spans restriction metalanguage in Appendix B. In addition, some are described in detail in Sections 6.6 and 6.7. Rather than go into detail about them here, only a few of each type will be described.

1. Positional restrictions: The BNF definitions are often a bit too large, indicating as they do only the broad outlines of a construction. Thus, restrictions are needed

to limit these definitions.

For example, from the BNF component alone, the following could occur: The noun variant flanked by its left and right adjuncts is given by the production

<LNR> → <LN> <NVAR> <RN>

The right adjunct may be null and the noun itself may sometimes be omitted. In the production

<LN> → <TPOS> <QPOS> <APOS> <NSPOS> <NPOS> .

all of <QPOS>,<APOS>,<NSPOS>, and <NPOS> may be null, and the article position <TPOS> may be filled by the article "the". This would then allow such a sentence as

"The is ... "

To prevent this there is a restriction which says that before <NVAR> can have a null value, the left adjunct <LN> must be checked to see if it takes one of the permissible values for a zeroed noun, e.g. "the best", "five", "these", "John's", but not "the".

There are a number of restrictions governing which subclasses of adverbs are acceptable in which positions. Thus, "generally" is an acceptable sentence adjunct while "very" is not.

Positional restrictions also control the zeroing of elements in a relative clause, making sure for example one and only one element in an assertion is zeroed.

A very important and time-saving restriction checks verbs against their possible objects. There are nearly sixty

alternatives for <OBJECT> but very few of these can occur with any given verb. With every verb in the word dictionary there is stored an attribute called OBJLIST which lists valid object alternatives for that verb. During the top-down parsing, matching is attempted only with those alternatives.

2. Strong syntactic restrictions: These are genuine grammatical restrictions, like number and case agreement. One such restriction says that if <NVAR> is singular then the core of <TPOS> is not plural, thus precluding such a phrase as

"these book" .

Another states that if <SUBJECT> takes a pronoun as its value, that pronoun is in the nominative case (except in special cosntructions), precluding the sentence

"Me and him bought the books".

3. Weak syntactic restrictions: These are also genuine grammatical restrictions, but not as serious in the sense that one would not be considered quite as illiterate making weak syntactic errors as strong syntactic ones. In this category is the restriction which says a certain type of noun (count nouns) requires something in the article position. For example, the parser rejected the sentence

"Enzyme preparations from rat and man were analyzed."

because in scientific English in general, which the lexicon reflects, the word "rat" is a count noun, even though in the sublanguage of pharmacology it is not a count noun.

4. Selectional restrictions: These restrictions check for attribute agreement and verge on the semantic. The sentence

"Skyscrapers leap."

which is perhaps acceptable in fairy tales or in some awkward metaphorical sense, would be rejected because a restriction would check the word dictionary under "leap" for a list of the attributes its subject may not take, e.g. inanimate. This check is made not only for the Subject - Verb construction, but also for transformations of that construction into a noun phrase. Thus the sentence

"Leaping skyscrapers can be dangerous."

would have the unambiguous interpreation. "It is dangerous for one to leap skyscrapers,"

5. Punctuation restrictions: In this category are various comma restrictions, for example, that a comma cannot occur between a verb and its immediately preceding subject. Also in this category is the restriction requiring a question to be followed by a question mark.

6. Optimizations: Some restrictions are optimizations for the top-down back-up parser. For example, at <OBJECT> before taking the alternative <TOVO>, a restriction looks ahead to make sure there is a "to" in the sentence. Similarly, matching with some constructions is not attempted unless there are enough words remaining in the sentence to complete it. While these restrictions aid in the parsing, they do not properly belong in a formal grammar of English.

7. Minword restrictions: When ambiguous sentences are encountered, the top-down parser merely lists all possible parses. For example, in the classic sentence

"They are flying planes."

one parse would have

<OBJECT> → <VINGO> → <*VING><OBJECT>

"flying planes"

and the second parse would have

<OBJECT> → <NSTGO> → <LN><NVAR>

"flying planes"

The options are ordered so that hopefully the preferred parse comes up first, and in fact this generally seems to be the case.

However, certain ambiguities are suppressed, namely what may be called permanent predictable ambiguities. Ambiguity like that in the sentence,

"We hid the flashlight in the kitchen under the sink."

will result whenever there is a sequence of prepositional phrases. In these cases, all parses except the first are suppressed by what are called the "Minword" restrictions. They are cosmetic in purpose and cause information to be lost. Ideally we would like our analysis to identify the prepositional phrases and indicate by pointers the slots in the larger strings into which they can fit. This, in fact, is one of the reasons for developing a nodal spans parser, which detects and stores such information efficiently.

This classification of the restrictions suggests the algorithmic approach to semigrammaticality developed in Chapter 7. Assume optimizations and minword restrictions have been eliminated. The analysis of a sentence would proceed as follows: First the parser would run with all the restrictions in effect. Then the classes of restrictions would be lifted one by one, in the following order:

1. Punctuation -- since no one should be held too responsible for his punctuation.
2. Selectional -- to allow the author some metaphors, awkward as they may be.
3. Weak syntactic -- in involved sentences, especially in scientific writing where one aims for conciseness, these may even be honest mistakes.
4. Strong syntactic -- if the 18th word is the subject and 87th word the verb, the author may have forgotten what his subject was by the time he reaches the verb.

Positional restrictions would not be lifted at all since that would open the door to nonsense strings of words.

2.4. Locating Relations

The restrictions for the top-down parser are defined in terms of the locating relations which themselves are written in the sublanguage of English and are compiled into stacks of primary routines which are FORTRAN subroutines in the parsing program.

The problem with grammatical relationships is not

stating them but finding the appropriate arguments within the sentence. The basic assumption, which solves this problem and makes it possible to cope with the unboundedly context-sensitive nature of natural language, is that there is really no "action at a distance" but rather that grammatically related elements are adjacent at some point in the tree. For any two words which have any grammatical relationship whatsoever, that relationship is expressible by a locating relation or a product of locating relations. Most remarkable is the small size of the set of locating relations (there are 25-30).

The bulk of the detailed definitions of these relations are dependent on the vagaries of the grammar, but their meaning is more fundamental. The most important of them are given below by the class into which they fall.

1. Basic tree-traversing relations:

Acting horizontally is

COEL which searches to the right and left for the named coelement in the string.

Acting vertically are

ELEM which looks down ← inverse→ IMMSTG which looks up until for the specified element. it encounters a string, & ELEMOF which looks up for the specified element.

CORERT goes all the way down the tree, without passing through adjunct nodes, until it encounters an atomic

or a string. This is the basic routine of the grammar, because typically we wish to check attributes of the cores of adjacent elements in a string.

2. Adjunct relations:

a. Left and Right Adjuncts:

LADJ locates the left adjunct ←inverse→ HOSTRT finds the core of the host of a left or right adjunct.

RADJ locates the right adjunct

b. Sentence adjuncts:

PRESA looks left for <SA> ←INVERSE→ HOSTSTRING finds the string in which <SA> is embedded, e.g. <ASSERTION>.

POSTSA looks right for <SA>

3. Subject - Verb - Object-of-be relations:

DEEPEST-VERB travels diagonally down the tree through verbal object strings to find the deepest verb. ←inverse→ ULTIMATE-SUBJECT travels up to find the subject when called at the deepest verb.

For example, in the sentence

"I have given up trying to learn to be able to understand this grammar."

the deepest verb is "understand" and its ultimate subject is "I". DEEPEST-OBJBE is a composite relation which first locates the deepest verb and if it is a verb like "to be", finds the object-of-be.

In the bottom-up parsing the notion of "core" becomes crucial and in fact is extended in meaning. The other locating relations cannot be used, however. Because of the merging that takes place in the nodal spans algorithm, it may not be possible to determine, for example, the left adjunct of a noun uniquely.

2.5. The Word Dictionary

The word dictionary lists for each word its possible grammatical categories (noun, tensed verb, etc.) and its attributes under each category. As an example, let us look at the dictionary entry for the word "face".

FACE is first of all a noun, and as a noun it has the attributes

SINGULAR since it can occur in the context

"This ____ is ..." but not in "These ___ are ..."

NCOUNT1 (count noun) since "face" can appear in the context

"The ____ is ..." but not in "___ is ..."

NONHUMAN since "face" cannot appear in the context

"The ____ { reads / believes / knows } ."

FACE is also a tensed verb, and as a tensed verb it is PLURAL since it cannot occur with a definitely singular subject.

FACE is also an (untensed) verb (as in "to face"). As a verb and a tensed verb it has the following attributes:

OBJLIST (alternatives of <OBJECT> the verb can govern):

DP1PN (adverbial preposition - preposition - noun) where the adverbial preposition is "up" and the preposition is "to", giving us the phrase "face up to".

NSTGO -- a noun string as in "We face severe budget cuts."

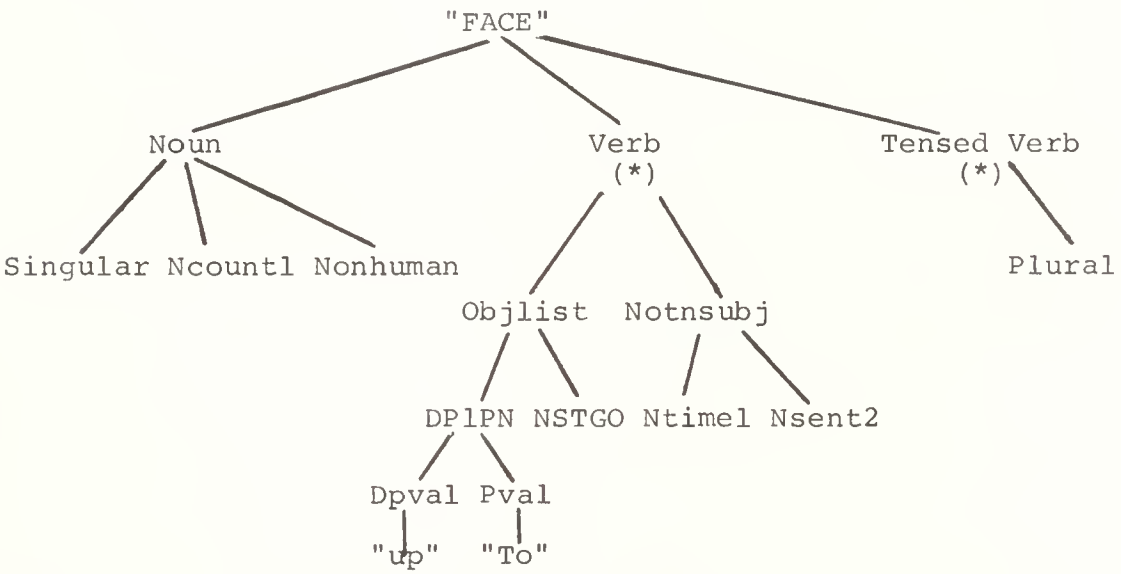
NOTNSUBJ (attributes its subject cannot take):

NTIME1 (time noun): ∄"Hours face forward."

NSENT2 (container noun):∄"The evidence seems to face forward."

Note that all these attributes are defined by their possible presence in certain test constructions, a standard technique of descriptive linguistics.

Word dictionary items are stored as tree structures. Thus, the definition of "face" could be represented as follows:



(*) A system of abbreviations makes it unnecessary to repeat portions of the tree.

CHAPTER 3. THE NODAL SPANS ALGORITHM

3.1. Informal Description.

Suppose we are given the inherently ambiguous grammar with productions

$$A \rightarrow B \ / \ C$$

$$B \rightarrow a$$

$$C \rightarrow a$$

where A, B, and C are nonterminal symbols and a is a terminal symbol, and suppose we are parsing the string

" a " .

A single-track backtracking bottom-up parsing algorithm will produce the two parse trees of Fig. 1.

Figure 1

A naive multi-tracking bottom-up parsing algorithm will produce the tree-structure of Figure 2, where the "$\vee$" link indicates an ambiguous parent node.

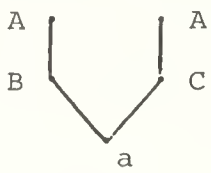


Figure 2

It is easy to see that for larger inherently ambiguous grammars with the first algorithm the time required will grow exponentially with the length of the string being parsed, and with the second algorithm both time and space requirements will grow exponentially. The nodal spans algorithm cuts the growth down to n^3 by merging identical nodes spanning the same substring. Thus, in the example, the two "A" nodes of Figure 2 would be merged, giving the tree-structure of Figure 3, where the "$\wedge$" link indicates an ambiguous offspring node.

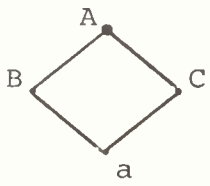


Figure 3

3.2. Formal Description.

In what follows, upper case latin letters near the beginning of the alphabet, A,B,C,..., represent non-terminal or terminal symbols of a grammar; the letter S represents the root symbol; lower case latin letters near the beginning of the alphabet, $a, a_1, a_2, \ldots, b, c, \ldots$, represent terminal symbols; upper case latin letters near the end of the alphabet, X,Y,Z,..., represent strings of terminals and non-terminals, and lower case letters in the middle of the alphabet, i,j,k,m,n,p,q,r, represent integers. Productions of a grammar will be indicated by a right arrow, "X → Y".

We will write $XY_1Z \Rightarrow XY_2Z$ if and only if $Y_1 \rightarrow Y_2$ is a production of the grammar. We will write $X_1 \overset{*}{\Rightarrow} X_n$ if and only if there are strings $X_2, X_3, \ldots, X_{n-1}$, such that $X_i \Rightarrow X_{i+1}$, $1 \leq i < n$. By convention, we will say $a \overset{*}{\Rightarrow} a$, $A \overset{*}{\Rightarrow} A$.

Let G be a context-free grammar in which every production is of the form

$$A \rightarrow X , \quad \text{where} \quad 0 < |X| \leq 2 . \tag{1}$$

That is, the right side of every production has either one or two terminal or nonterminal symbols. This is a weaker condition on G than that G is in Chomsky normal form, and the proof that any context-free grammar can be put into form (1) mimics the proof that any context-free grammar can be put into Chomsky normal form.[1] Let

$$a_1 \, a_2 \, \cdots \, a_n$$

be a string which is to be parsed according to grammar G.

The nodal spans parsing process consists in building up a list of what are called nodal spans.

Definition: A nodal span is a quadruple

$$\langle p, A, q \; D \rangle \tag{2}$$

where p and q are integers, $0 \leq p < q \leq n$; A is a nonterminal or terminal symbol such that

---

[1] See for example Noam Chomsky, "On Certain Formal Properties of Grammars," Information and Control, Vol. 2, p. 149 or Hopcroft and Ullman, Formal Languages and their Relation to Automata, Reading, Mass., 1969, p. 51.

$$A \overset{*}{\Rightarrow} a_{p+1}\ a_{p+2} \cdots a_q\ ,$$

i.e. the symbol A ultimately derives the substring $a_{p+1}, \cdots a_q$; and $\mathcal{D}$, a possibly empty set called the divisions list of the span, is defined as follows:

Definition: The <u>divisions list</u> $\mathcal{D}$ of the span (2) is the set of

(a) triples $\langle B,C,r \rangle$

where B and C are nonterminal or terminal symbols and r is an integer, $p < r < q$, such that

$$A \rightarrow B\ C$$

is a production in the grammar,

$$B \overset{*}{\Rightarrow} a_{p+1} \cdots a_r\ ,$$

and

$$C \overset{*}{\Rightarrow} a_{r+1} \cdots a_q\ ;$$

and

(b) singletons $\langle D \rangle$

where D is a nonterminal or terminal symbol such that

$$A \rightarrow D$$

is a production in the grammar, and

$$D \overset{*}{\Rightarrow} a_{p+1} \cdots a_q\ .$$

Alternatively, we may think of the divisions list $\mathcal{D}$ of span (2) as a list of pointers to the possible offpsring nodes of node A.

Note that the divisions list for a span (2) is empty

if and only if A is a terminal symbol.

Clearly the string is an acceptable sentence for grammar G if and only if the nodal span

$$<0,S,n,\mathcal{D}>$$

occurs in the list of spans.

There are three steps in the nodal spans parsing algorithm:

Step (1) The input symbols are read in a left-to-right manner. When the input symbol (i.e. terminal symbol) a_i is read, the span

$$<i-1,a_i,i,\phi>$$

is formed and added to the list of nodal spans.

Step (2) For every nodal span $<p,D,q,\mathcal{D}>$, and for every production $A \rightarrow D$, the quadruple

$$<p,A,q,\{<D>\}> \quad . \tag{3}$$

is formed. If there is a nodal span of the form

$$<p,A,q,\mathcal{D}'>$$

already on the list, the element $<D>$ is added to the set $\mathcal{D}'$. Otherwise, span (3) is added to the list of nodal spans.

Step (3) For every nodal span $<r,C,q,\mathcal{D}>$ and for every production $A \rightarrow B\ C$ such that a span of the form

$$<p,B,r,\mathcal{D}'>$$

is already on the list of nodal spans, the quadruple

$$<p,A,q,\{<B,C,r>\}> \quad . \tag{4}$$

is formed. If there is a nodal span of the form

$$\langle p,A,q,\mathcal{D}' \rangle$$

already on the list, the element <B,C,r> is added to the set $\mathcal{D}'$. Otherwise span (4) is added to the list of nodal spans.

The spans may be processed sequentially by placing them on a first-in, first-out stack as they are formed, or recursively by placing them on a first-in, last-out stack as they are formed.

It is easily shown that a quadruple is generated by the above algorithm if and only if it is a valid nodal span.

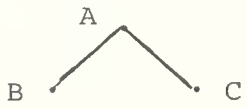
A particular parse tree can be determined from the collection of nodal spans as follows: Begin with the span $\langle 0,S,n,\mathcal{D}_1 \rangle$; if this span does not occur, the string is not a sentence and has no parse tree. The root node of the parse tree is labelled S and the span is placed on a to-be-processed list. For any nodal span

$$\langle p,A,q,\mathcal{D}_i \rangle \tag{5}$$

on the to-be-processed list, pick any element from $\mathcal{D}_i$. If this element is a singleton <D>, then give the node labelled A in the parse tree a single offspring node labelled D

and place the span $\langle p,D,q,\mathcal{D}_j \rangle$ on the to-be-processed list.

If the selected element is a triple <B,C,r>, then give the node labelled A in the parse tree, two offspring nodes labelled B and C,



and place the spans <p,B,r,$\mathcal{D}_k$> and <r,C,q,$\mathcal{D}_\ell$> on the to-be-procsssed list. If $\mathcal{D}_i$ is empty, A is a terminal node and has no offspring nodes. Remove span (5) from the to-be-processed list. Continue this procedure until the to-be-processed list is exhausted.

It is easily shown that all parse trees can be found in this way and that all trees found in this way are valid parse trees. The sentence is ambiguous if and only if some $\mathcal{D}_i$ encountered in this procedure has more than one element.

The particular algorithms used for testing are specified more precisely in the algorithm-specification language SETL in Sections 4.3 and 4.4.

3.3. Example.

As an example, let us look at the fragmentary grammar

<ASSERTION> → <NSTG> <VO>

<VO> → <*TV> <OBJECT>

<OBJECT> → <NSTG> / <VINGO>

<NSTG> → <APOS> <*N> / <*N> / <*PRO>

<APOS> → <*ADJ> / <*VING>

<VINGO> → <*VING> <OBJECT>

Let us see how the algorithm would proceed in parsing the sentence

"They are starving dogs."

according to this grammar. In this example we will consider the divisions lists to be sets of pointers to their offspring nodes, indicated by the number to the left of the span.

The lexical scanner would present us with the string

<*PRO> <*TV> <*VING> <*N>

The first token "<*PRO>" is read, and the span

1. <0, <*PRO>, 1, ϕ>

is formed. By step (2) of the algorithm the spans

2. <0,<NSTG>,1,{(1)}>

3. <0,<OBJECT>,1,{(2)}>

are formed. The second token "<*TV>" is read, and the span

4. <1,<*TV>,2,ϕ>

is formed. Since <*TV> does not appear in any singleton production nor as the second element in any binary production, no more spans are formed at this point. The third token <*VING> is read, and the span

5. <2,<*VING>,3,ϕ>

is formed, and then by step (2) the span

6. <2,<APOS>,3,{(5)}> .

The fourth token <*N> is read and the spans

7. <3,<*N>,4,ϕ>

8. <3,<NSTG>,4,{(7)}>

are formed. Now <*N> appears as the second element in the production

<NSTG> → <APOS> <*N> .

Hence, according to step (3) of the algorithm we look for a span whose symbol is <APOS> and whose endpoint is 3, and find that span 6 is such a span. Therefore we form the new span

9. <2,<NSTG>,4,{(6,7)}> .

This completes the processing of span 7. Span 8 gives rise to the span

10. {3,<OBJECT>,4,{(8)}>

via step (2), and similarly span 9 gives rise to the span

11. <2,<OBJECT>,4,{(9)}>.

In processing span 10, it is seen that <OBJECT> is the second element of two different productions:

<VO> → <*TV> <OBJECT>

and

<VINGO> → <*VING> <OBJECT>

Looking therefore for a span whose endpoint is 3 and whose symbol is <*TV> or <*VING> we find span 5, and by step (3) form the new span

12. <2,<VINGO>,4,{(5,10)}>.

On the other hand, while processing span 11, we look for a span with symbol <*TV> or <*VING> and endpoint 2, find span 4,

and form the span

13. <1,<VO>,4,{(4,11)}>.

Span 12 gives rise via step (2) to a span with symbol <OBJECT>, initial point 2 and endpoint 4, but we already have such a span, viz. span 11. Therefore we merely add to the divisions list of span 11:

11. <2,<OBJECT>,4,{(9),(12)}>.

This change has no effect on the node already lying above <OBJECT> in the tree and does not mean span 11 has to be processed again. Finally, span 13 is processed; <VO> is found to be the second element in the production

<ASSERTION> → <NSTG> <VO>;

and is coalesced with span 2 to form the new span

14. <0,<ASSERTION>,4,{(2,13)}>,

completing the parse successfully.

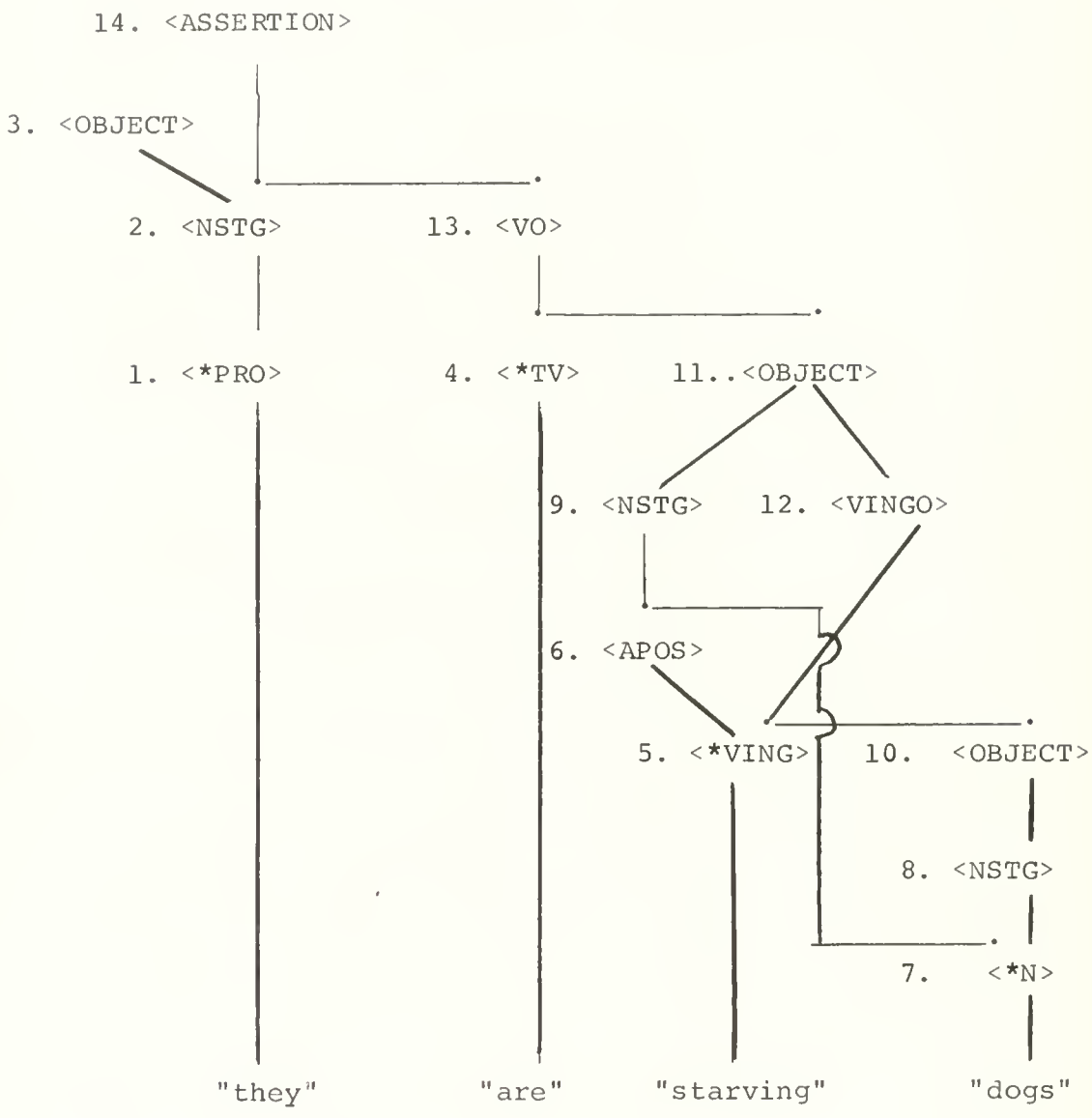
The structure of the span collection built up by the parsing algorithm is shown in Figure 4.

Note that the spans were processed on a first-in, first-out basis. They could also have been processed on a first-in, last-out basis in a recursive environment.

The final parses of the sentence can be found by starting at the root span (span 14 for <ASSERTION>) and following down the divisions lists as described in the previous section. Any node which has a divisions list with more than one element (such as span 11 for <OBJECT>) spans an ambiguous portion of the sentence. The ambiguity

is represented succinctly in the span collection generated by the algorithm.

Figure 4



The ⌊ and ⌐ links represent coelement or sibling relations; the ⟋⟍ links represent ambiguous offspring nodes; and the ⟍⟋ links represent ambiguous parent nodes.

D. Candidates Lists

Not all the nodes generated by the algorithm have a place in the final parse tree. In the example, only node 3 for <OBJECT> does not take part in a final parse of the sentence, but this is because the grammar we used was very sparse. In parsing a sentence with a reasonably full English grammar, there will very likely be a large number of irrelevant nodes generated by the algorithm because of the high degree of local ambiguity in English.

A method for cutting down on the number of irrelevant nodes was introduced by Jay Earley[2] and will be explained here as the device of candidates lists. Notice in the above example that although the symbol <NSTG> can be derived from <OBJECT> there is no way <OBJECT> can span the first portion of a sentence. Therefore, we could have avoided generating span 3 if we had restricted ourselves at that point to symbols which can appear on the leftmost branch of a tree depending from the root symbol <ASSERTION>. Furthermore, we can cut down on the number of irrelevant nodes generated in parsing the latter portions by generating only nodes which are consistent with the portion of the sentence already scanned. Thus, if we encountered the word

" ... face ... "

in a sentence, it could be interpreted as a tensed verb and

---

[2] Earley, Jay, An Efficient Context-Free Parsing Algorithm, Ph.D. Thesis, Carnegie-Mellon University, 1968.

the appropriate nodes could be generated. If however the previous word was "the", it is impossible for "face" to be a tensed verb and we could avoid generating all the spans which assume it is. Of course, all irrelevant spans would be eliminated in this way only if the interpretation of a token were determined uniquely by its left context, and it is highly unlikely that such a grammar could be constructed for English.

More formally, we can define for any symbol A the set

$$F(A) = \{B \mid (\exists X)\ (A \overset{*}{\Rightarrow} B\ X)\}\ .$$

Intuitively, a symbol B is in F(A) if and only if it can be first or leftmost symbol in a tree depending from A. In the sample grammar of the above example,

$$F(\langle ASSERTION\rangle) = \{\langle ASSERTION\rangle, \langle NSTG\rangle, \langle APOS\rangle, \langle *N\rangle, \langle *PRO\rangle, \langle *ADJ\rangle, \langle *VING\rangle\},$$

$$F(\langle VINGO\rangle) = \{\langle VINGO\rangle, \langle *VING\rangle\},$$

$$F(\langle *TV\rangle) = \{\langle *TV\rangle\}.$$

The candidates lists are formed during the parsing and there is one for each input symbol. The candidates list C_i for a symbol a_i in the input string lists all the symbols which could span a substring beginning at a_i consistent with the parsing of $a_1 \ldots a_{i-1}$. In particular,

$C_0 = F(S)$, where S is the root symbol of the grammar.

$C_{i+1} = \{D \mid (\exists p,A,B,C)$ (the span $\langle p,B,i,\mathcal{D}\rangle$ has been generated and $A \rightarrow B\ C$ is a production in the grammar and $D \in F(C))\}$.

We can augment the basic algorithm to build up the candidates lists during the parsing by inserting the following steps:

Step (0) Set C_0 equal to $F(S)$, S the root node. Initialize C_i to $\emptyset$, $1 \leq i \leq n$.

In steps (1), (2), and (3), add a quadruple $\langle p,A,q,\mathcal{D}\rangle$ to the list of nodal spans only if the symbol A appears on the candidates list C_{p+1}.

Step (4) When any span $\langle p,B,q,\mathcal{D}\rangle$ is generated, look for productions of the form

$$A \rightarrow B\ C$$

and for each such production perform the operation

$$C_{q+1} = C_{q+1} \cup F(C) .$$

The effect of using candidates lists in parsing English will be examined in Chapter 4.

3.5. Space and Time Requirements

The following terminology is used in this section:

p	=	the initial point of a span;
q	=	the end point of a span;
r	=	the division point of a span;
A,B,C	=	symbols in the grammar, usually in connection with the production $A \rightarrow B\ C$;
n	=	number of symbols in the sentence of input string being parsed;
N	=	number of symbols in the grammar;

P = number of productions in the grammar ("$A \rightarrow X_1 \ / \ X_2 \ / \ X_3$" counts as three productions);

P_C = number of productions in grammar in which C appears on right side of a production;

P^A = number of productions in a grammar in which A appears on the left side of the production;

Σ = the set of nonterminal and terminal symbols in the grammar.

Number of Spans: Clearly for each p and q the number of spans is limited by N. For each q, there are q possible values of p. q can vary from 1 to n. Thus, the number of spans generated by the algorithm over an n-symbol input string is bounded by

$$\sum_{q=1}^{n} \sum_{p=0}^{q-1} N \approx \frac{1}{2} N n^2 .$$

Space: Space requirements are determined by the size of the divisions lists for each span. For a given p, q, and A, the number of elements in the divisions list $\mathcal{D}$ of the span $\langle p,A,q,\mathcal{D} \rangle$ is bounded by

$$P^A(q - p)$$

because A may divide into its constituents B and C at any one of the q - p points between a_{p+1} and a_q. Therefore the total space requirements for the algorithm for an n-symbol input string is bounded by

$$\sum_{q=1}^{n} \sum_{p=0}^{q-1} \sum_{A \in \Sigma} P^A(q - p) \approx \frac{1}{6} P n^3 .$$

Thus space requirements grow as n^3.

Time: The processing of each span <r,C,q> involves several steps:

(1) Finding all B such that

$$A \rightarrow B\ C$$

is a production in the grammar. This introduces a factor of

$$k_1\ P_C\ , \quad \text{for some constant } k_1 > 0\ .$$

(2) For each such B and for r, checking the (B,r) entry of an N × n matrix of pointers to heads of lists of spans of the form

$$\langle p,B,r \rangle .$$

This requires a constant time k_2.

(3) Link through the list of spans <p,B,r> and for each such span form the span

$$\langle p,A,q \rangle$$

This step is bounded by

$$k_3\ r\ , \quad \text{for some constant } k_3 > 0\ .$$

Thus the total time required for processing one span is bounded by

$$k_1\ P_C(k_2 + k_3 r)$$

The total time required for processing all spans generated during parsing, that is, the total time required for the parsing, is bounded by

$$\sum_{q=1}^{n} \sum_{r=0}^{q-1} \sum_{C \in \Sigma} k_1\ P_C(k_2 + k_3\ r) \approx \frac{1}{3}\ K\ P\ n^3\ ,$$

$$\text{for some constant } K > 0.$$

Thus time also grows as n^3.

3.6. History

As noted before, the nodal spans parsing algorithm in its earliest formulation is due to John Cocke. It was developed independently by Younger[3] who describes the algorithm in terms of a three-dimensional binary recognition matrix in which the (i,j,k)-th entry is equal to one whenever the kth symbol in the grammar spans the substring i symbols in length starting from the jth input symbol; the divisions lists are recaptured from the grammar. Kasami and Torii[4] give a version of the algorithm for nonambiguous grammars which is substantially the same. Earley[5] gives a detailed version of the algorithm together with complexity results. He does not require the grammar be given in Chomsky normal form but rather treats it internally as though it were. He defines states which essentially specify initial and end points of a span and the nonterminal symbol, and performs three operations on them -- predictor, which corresponds to the device of candidates lists; completer, which corresponds to step (3) of the algorithm given above; and scanner, which corresponds to step (1) of the algorithm. In addition he incorporates k-symbol lookahead. The description of the algorithm given here follows Cocke and Schwartz.[6]

---

[3] Younger, D. H., "Recognition and Parsing of Context-Free Languages in Time n^3," Inf. & Control, 10, 2 (2/67) 189-208.

[4] Kasami, T. & Torii, K., "A Syntax-Analysis Procedure for Unambiguous Context-Free Grammars," JACM, 16, 3 7/69, 423-431.

[5] Earley, Op. Cit.

[6] Cocke, J. & Schwartz, J., Compilers & Compiler Writing Techniques, New York University, 1969.

CHAPTER 4. NODAL SPANS PARSING WITHOUT RESTRICTIONS: STATISTICAL RESULTS

4.1. Introduction

Chapter 4 describes studies which were made using only the context-free component of the String Project's grammar, without invoking restrictions. The first goal was to test the nodal spans algorithm on a large context-free grammar in order to gain information about the practical efficiency of the algorithm. Secondly, it was once hoped that a nodal spans parse might be the first pass in a parsing process which would use the results of this first pass as an optimization in a top-down back-up second pass during which the restrictions would be called. This did not turn out to be practical, and in fact, the principal result of these studies is to point out the importance of the restrictions in the grammar.

Given a grammar which is not in Chomsky normal form, i.e. in which there are productions whose right sides are longer than two symbols, as in (1), there are two approaches that may be taken.

$$A \rightarrow B\ C\ D\ \ldots\ E \qquad (1)$$

The grammar may be treated as given, thus producing derivation trees as in Figure 1:

Figure 1

Or the grammar may be treated internally as though it were binary by leftward grouping, producing the derivation tree of Figure 2.

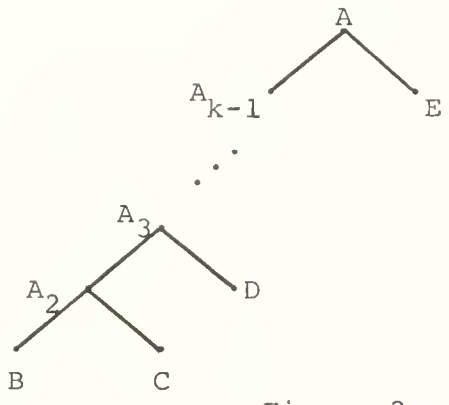


Figure 2

(Here and throughout, A_i indicates that the first i symbols of an expansion of A have been matched.) The second approach may be called "merging". In the algorithm with merging the growth of time and space requirements is bounded by n^3, whereas without merging the bound is n^{k+1} where k is the length of the longest production.

The difficulty with the use of merging in natural language parsing is that restrictions are harder to execute. In the nodal spans algorithm the restrictions either cannot look more than one level below the node at which they are called or else may at some time have to restructure parts of the span collection already built. In the algorithm with merging, non-adjacent coelements in a string appear on different levels of the tree. Yet many restrictions operate on coelements which are not adjacent. For example, in the string for <ASSERTION>

<SA><SUBJECT><SA><VERB><SA><OBJECT><RV><SA>

there is a restriction requiring agreement between the object and the subject if the verb is a form of "to be":

"That man is a banker."

"These men are bankers."

Yet with merging, the subject is no longer accessible when the object is encountered. Another restriction requiring agreement between distant coelements in the <ASSERTION> string is the one involving reflexive pronouns in the final sentence adjunct position:

"I built the cabin myself."

"He built the cabin himself."

Such restrictions are easier to state and execute if no merging takes place.

Therefore the question that is asked in these studies is, "Does the algorithm without merging really blow up disastrously?" The answer turns out to be an emphatic "Yes!" and this leads to a second question: "Does the algorithm with merging do any better?" The answer is "Substantially, but not enough." These results dictate the form of the nodal spans algorithm which must be used in natural language parsing and influence very profoundly the structure of the restriction metalanguage developed in Chapter 5.

The grammar used in these tests is listed in Appendix D and its form explained in Section 2. In Sections 3 and 4 the nonmerging and merging algorithms and associated data

structures are specified. In Section 5 the r sults of the statistical tests are presented. Comparisons are made between the algorithms without candidates lists and with candidates lists.

4.2. The Inverse Grammar.

In the remainder of this work the grammar is assumed to have the following form: There is one production for every nonterminal symbol A,

A → X / Y / ... / Z

where X,Y,...,Z are non-null strings of nonterminal and terminal symbols which constitute possible expansions or alternatives of the symbol A. The strings may be of any length greater than or equal to one; no null productions are allowed. A string X may have the form

[B] C [D] ... E ,

where B,...,E are nonterminal or terminal symbols, any of which may be optional, indicated by the brackets [B]. Every context-free grammar can be put into this form. Briefly, we may rewrite the productions

A → B C D

B → E F / null

as

A → [B] C D

B → E F

It will be useful to distinguish several types of elements in a string. An element in a string is <u>possibly initial</u> if it is preceded only by optional elements. Thus in the string

[B] [C] D [E] (2)

B, C, and D are possibly initial and E is not. An element is necessarily initial if it is the first element in the string. Thus only B is necessarily initial in (2). An element is possibly terminal if it is followed only by optional elements. Thus, in (2) D and E are possibly terminal. The possible successors of a symbol are those elements which follow it in the string, up to and including the first nonoptional element. Thus in (2), B has C and D as possible successors, C has only D, D has E, and E has no possible successors.

The location of a symbol B in a production will be indicated by an ordered triple

<A,k,j>

where A is the nonterminal symbol in whose production B is the jth element of the kth alternative. For example, in the production

A → X / Y / B C D / Z

the location of C is indicated by the triple <A,3,2>. Of course an element may occur more than once on the right side of the same production.

In the SETL programs below, the grammar is stored in inverse form in an array called ref. The element ref(symbol) is a set of entries, one for each occurrence of the symbol on the right side of a production. Each entry contains, in addition to the location <A,k,j>, of the occurrence, flags

indicating whether or not the symbol is possibly initial, necessarily initial, and possibly terminal in that occurrence. Furthermore, each entry contains a list of the symbol's possible successors; this is used in construction of the candidates lists. In particular, the following functions extract information from the entries:

eta(entry) = the A of the location triple above.

kay(entry) = the k of the location triple above.

jay(entry) = the j of the location triple above.

posinitflag(entry) = t if the symbol is possibly initial, f otherwise.

necinitflag(entry) = t if the symbol is necessarily initial, f otherwise.

terflag(entry) = t if the symbol is possibly terminal, f otherwise.

possuc(entry) = a list of the symbol's possible successors in that occurrence.

jayl(entry) = the number of elements to the left of the symbol up to and including the first nonoptional element.

4.3. The Nonmerging Parsing Algorithm.

In the algorithm of this section, incomplete spans are formed as a computational aid, but no merging takes place in the middle of a string. Incomplete spans which are identical except that their descendant spans differ are listed as different incomplete spans. Although concatenation

takes place between an incomplete span I_1 on the left and a complete span K on the right, the list of descendants of the new incomplete span I_2 is a list of the descendants of the old incomplete span I_1 together with the complete span K.

Let

$$A \rightarrow B\ C\ D\ \ldots\ E$$

be a production in the grammar. When the complete span $\langle r_2,D,r_3\rangle$ is generated, a search is made for incomplete spans of the form $\langle p,A_2,r_2\rangle$. For each such span found, a new incomplete span $\langle p,A_3,r_3\rangle$ is formed. If the descendants of $\langle p,A_2,r_2\rangle$ are $\langle p,B,r_1\rangle$, $\langle r_1,C,r_2\rangle$, then the descendants of $\langle p,A_3,r_3\rangle$ are $\langle p,B,r_1\rangle$, $\langle r_1,C,r_2\rangle$, $\langle r_2,D,r_3\rangle$.

Merging takes place when the complete expansion of A is recognized. Then only one complete span is formed spanning the one substring, and all of its possible sets of immediate descendants are listed in its divisions list.

In the SETL specification below the incomplete spans are stored in the set ispans and the list of immediate descendants of an incomplete span is returned as the value of the function idivlis. The complete spans are stored in the set cspans and the set of possible lists of immediate descendants of a complete span is returned as the value of the function cdivlis.

In the algorithm, a complete span is generated for each category or "part of speech" a word can assume. These complete

spans are placed on stack. Every span on stack is added to cspans if it is not already there; otherwise the divisions list of the span already there is augmented. If the span is new, then for each of the occurrences of its symbol in a possibly initial position in a string, an incomplete span is formed, and for every occurrence which is not necessarily initial, all possible concatenations with incomplete spans are searched for and the appropriate new incomplete spans are generated. For every incomplete span which has matched all but the optional elements of a production, a complte span is generated and placed on stack.

sentence is an sn-tuple representing the sn words of the sentence to be parsed. Each word is a set of categories. ref is the inverse grammar and root its root symbol. eta, kay, and jay of a grammar entry are explained in the previous section. A complete span is a triple <p,A,q> where

pee(cspan) = p, the initial point of the span,

eta(cspan) = A, the span's symbol,

and que(cspan) = q, the end point of the span.

An incomplete span is a triple <p,<A,k,j>,q> where pee and que are as above,

eta(ispan) = A, the span's symbol,

and kay(ispan) = k, and jay(ispan) = j, indicate that the first j symbols in the kth alternative of A have been matched.

The device of the candidates lists is activated by the switch candswitch. The F-relationship is given by the function

first(symbol) = the set of all symbols which can appear leftmost on a tree depending from symbol.

The sets,

candlist(k) = the set of all symbols which can span a substring of the input string beginning at the (k+1)st word,

are augmented each time an incomplte span is formed. Before a complete span <p,A,q> is generated, A is checked against candlist(p).

```
define nonmergeparse;
/*the following variables, sets and functions are assumed
  to be global: sentence, sn, ref, root, eta, kay, jay,
  jayl, posinitflg, necinitflg, posterflg, possuc, pee,
  que, candswitch, first */
if candswitch then candlist(0) = first(root);
(1 ≤ ∀i ≤ sn) word = sentence(i);
  /*stack up all meanings of the ith word */
  (∀cat ∈ word) if n candswitch or cat ∈ candlist(i-1) then
    stack = stack with (<i-1,cat,i> is x); cdivlis(x)=nℓ;
    end ∀cat;
  /*for every span on the stack, see if it has been
    generated already */
```

```
  (∀cspan ∈ stack doing stack=stack less cspan;)
    if (∃csp∈cspans|(∀f∈{pee,que,eta}) f(csp) eq f(cspan))
    /*if so, modify the divisions list of the existing span */
    then cdivlis(csp)=cdivlis(csp) + cdivlis(cspan);
    /* if not, add the new span to cspans */
    else cspans = cspans with cspan;
      (∀occ ∈ ref(eta(cspan)))
      /*for every possibly initial occurrence of the symbol
        in the grammar, generate a new incomplete span */
      if posinitflg(occ) then if n candswitch or eta(occ) ∈
          candlist(pee(cspan))
        then iproc(<pee(cspan),<eta(occ),kay(occ),jay(occ)>,
                    que(cspan)>,<cspan>);
          if necinitflg(occ) then continue ∀occ;
      /*for every occurrence of the symbol in the middle
        of a production, search for a matching incomplete span*/
      (∀isp∈ispans|(eta(occ) is et)eq eta(isp) and(kay(occ)is k)
        eq kay(isp) and (jay(occ) is j)-jayl(occ) le jay(isp)
        and jay(isp) le j-1 and pee(cspan) eq que(isp))
        isproc(<pee(isp),<et,k,j>,que(cspan)>,idivlis(isp)
                    + <cspan>);
      end ∀isp;  end ∀occ;;
end ∀cspan; end ∀i;  end  nonmergeparse;

macro isproc(ispan,div);
ispans = ispans with ispan;
idivlis(ispan) = div;
```

[continued]

```
/* if appropriate, form a complete span from the incomplete
     span */
if posterflg(occ) then stack = stack with (<pee(ispan),
     eta(ispan), que(ispan)> is x);  cdivlis(x)=idivlis(ispan);;
if candswitch then candlist(i) = candlist(i)
     + first[possuc(occ)];;
endm;
```

4.4. The Merging Parsing Algorithm.

In the merging algorithm, a span is the triple <p,<A,k,j>,q> like the incomplete span of the previous algorithm. If j is the length of the kth possible expansion of A, i.e. if the string has been completely matched, k and j are given the values 0, and the span is placed on stack as soon as it is generated. The spans with k = j = 0 correspond to the complete spans of the previous algorithm, the others to the incomplete spans. If a span represents the matching of all but optional elements in a string, it could be used as a complete span and placed on the stack or it could be used as an incomplete span seeking a match with the remaining optional elements. Therefore such a span causes another span to be generated with k = j = 0 and the given span as its immediate descendant. A span <p,<A,k,j>,q> formed by the concatenation of the spans <p,<A,k,j-1>,r> and <r,<B,0,0>,q> has in its divisions list the ordered pair

<<p,<A,k,j-1>,r>,<r,<B,0,0>,q>>. Thus each node has at most two immediate descendants.

The spans are stored in the set spans. Their divisions lists are returned as the values of the function divlis. Otherwise the sets and functions of this algorithm are the same as in the previous algorithm.

```
define mergeparse;
/* the following are assumed to be global: sentence, sn, ref,
     eta, root, kay, jay, jayl, posinitflg, necinitflg,
     posterflg, possuc, pee, que, candswitch, first */
if candswitch then candlist(0) = first(root);
(1 ≤ ∀i ≤ sn) word = sentence(i);
   /* stack up all meanings of the ith word */
   (∀cat∈word) if n candswitch or cat∈candlist(i-1) then
      stack=stack with (<i-1,<cat,0,0>,i> is x);
   divlis(x) = nl;  addspan(x);;  end ∀cat;
   /* for all spans on stack and all occurrences of symbol
     in grammar */
   (∀span ∈ stack doing stack = stack less span;)
   (∀occ ∈ ref(eta(span)))
   /* for every possibly initial occurrence of the symbol in
     the grammar, generate a new span */
   if posinitflg(occ)
   then if n candswitch or eta(occ)∈candlist(pee(span))
      then newsp=<pee(span),if possuc(occ) eq nl then
         <eta(occ),0,0>
         else <eta(occ),kay(occ),jay(occ)>, que(span)>;
```

```
      divlis(newsp) = {<span>}; addspan(newsp); termhandl;;
     if necinitflg(occ) then continue ∀occ;;;
   /* for every occurrence of a symbol in the middle of a
     production, search for a matching span */
   (∀sp∈spans|(eta(occ) is et) eq eta(sp) and (kay(occ) is k)
      eq kay(sp) and (jay(occ) is j) -jayl(occ) le jay(sp) and
      jay(sp) le j-1 and pee(span) eq que(sp))
   newsp =<pee(sp),if possuc(occ) eq nl then <et,0,0> else
         <et,k,j>, que(span)>;
   divlis(newsp) = {<sp,span>}; addspan(newsp); termhandl;
end ∀sp; end ∀occ;; end ∀span; end ∀i;  end mergeparse;

macro termhandl;
if(kay(newsp) eq 0 and jay(newsp) eq 0) then
     stack = stack with newsp;;
/* if the new span is possibly terminal but not necessarily
     terminal generate a necessarily terminal span and
     stack it */
if posterflg(occ) and possuc(occ) ne nl
then nusp = <pee(newsp),<eta(occ),0,0>, que(newsp)>;
   divlis(nusp) = {<newsp>}; addspan(nusp);
   stack = stack with nusp;;
/* augment the candidates list */
if candswitch then candlist(i) = candlist(i)+first[possuc(occ)];;
endm;
```

```
macro addspan(spn);
/* see if span has been generated already */
if (∃spa∈spans|(∀f∈{pee,que,eta,kay,jay}) f(spa) eq f(spn))
/* if so, modify the divisions list of the existing span */
then divlis(spa) = divlis(spa) + divlis(spn);
/* if not, add the new span to spans */
else spans = spans with spn;;
endm;
```

4.5. The Statistical Results.

The experiment: The merging and nonmerging algorithms were programmed in FORTRAN for the CDC 6600 at New York University. Because difficulties with space requirements were anticipated, the areas set aside for the spans and divisions lists generated were as large as possible, and the entire program occupied 320K 60-bit words of core. Furthermore information was packed tightly into the words.

Initial tests on the merging algorithm showed that it could parse sentences of up to 17 words, but the non-merging algorithm ran out of core after 9 words. It was therefore decided to run tests on a collection of 100 English sentences of 9 words or less. The sentences were taken from a pharmaceutical article and pared down to size. There were 20 sentences each of 5 words, 6 words, 7 words, 8 words, and 9 words. An effort was made to include all the major constructions in the grammar. Since the algorithm looks at no more than the category of a word, the sentences were input as strings of terminal symbols, thereby obviating a lexical span pass.

These sentences were parsed with both the merging and the nonmerging algorithms and for each of these with and

without candidates lists. Thus four different tests were run on the set of sentences. All of the sentences parsed, without overflow difficulties, with the merging algorithm both with and without candidates lists. With the nonmerging algorithm all sentences of seven or fewer words parsed both with and without candidates lists. Of the 20 eight-word sentences, all parsed with candidates lists, 18 parsed without. Of the 20 nine-word sentences, 11 parsed with candidates lists, and only 6 without.

As an indication of the difficulties that arise, we might look briefly at the parses found for the simplest two word sentence

<*N> <*TV> .

In addition to finding the correct parse, the parser also interprets the noun as being in the noun position <NPOS> of the left adjunct of another noun which is deleted (or "understood"). It is as if we shortened the noun phrase "house boat" to "house" just as we might shorten the noun phrase "the best boat" to "the best", "boat" being understood. Of course there is a restriction in the full grammar which prevents this sort of thing from happening. In longer sentences, the amount of such highly spurious ambiguity multiplies drastically.

Moreover, the exploding space requirements are unsurprising when one considers that we are taking a grammar which indicates the full broad range of English constructions

and stripping it of all the constraints which limit the application of these constructions. The grammar of 200 nonterminal symbols rests on a base of only about 20 terminal symbols, so the ballooning of possible interpretations in the middle of the span collection is to be expected.

Results for the Merging Algorithm: From Section 3.5 we know that for the merging algorithm the theoretical bound on the number of spans is Kn^2 where $K \leq \frac{1}{2} N$, N the number of symbols in the grammar. When the grammar used is treated as a binary grammar by the algorithm, there are 564 symbols, nonterminal, terminal, and literal, giving an upper bound of $282\ n^2$. If possible, we would like to know from the statistical studies whether the number of spans does grow as n^2 and if so, what the value of K really is. Similarly, the theoretical bound on the number of divisions lists is $K'n^3$, where $K' \leq \frac{1}{6} P$, P the number of production alternatives in the grammar. Treated as binary, the grammar has 770 production alternatives, giving an upper bound of $128\ n^3$. Again we would like to know if n^3 is in fact the rate of growth and if so what the value of K' is. Time requirements theoretically grow as n^3, and we would like to know if this is bourne out in the tests. Here we are not interested in the leading constant since it is dependent on the machine and vagaries of the programming.

The following table, which represents the averages over all nine-word sentences, should give some idea of the typical

rates of growth:

WORD:

1	2	3	4	5	6	7	8	9
Number of Spans: without Candidates Lists:								
107.1	307.2	606.1	1010.9	1546.4	2229.7	3044.7	4007.7	5094.4
with Candidates Lists:								
50.3	139.9	277.8	478.0	767.6	1166.0	1662.3	2272.6	2988.5
Number of Divisions Lists: without Candidates Lists:								
84.5	282.7	639.3	1207.3	2066.3	3302.0	4950.5	7127.8	9883.0
with Candidates Lists:								
35.8	114.5	259.5	512.2	930.3	1583.4	2511.6	3803.6	5513.5
Time Requirements: without Candidates Lists:								
0.078	0.298	0.743	1.527	2.819	4.838	7.747	11.844	17.309
with Candidates Lists:								
0.052	0.179	0.424	0.858	1.597	2.798	4.592	7.202	10.789

Least squares analysis was done on these results. On the assumption that the growth in number of spans is on the order of n^2, the leading terms of the approximations found were

Without Candidates Lists: $63.3\ n^2$

With Candidates Lists: $43.4\ n^2$

This suggests that without candidates lists, about 22% of the theoretically possible spans are actually generated. With candidates lists, only about 15%. While nine values are too few for adequate regression analysis, there are several indications that the growth is in fact quadratic.

The cubic approximation for the candidates lists case is asymptotically greater than that for the non-candidates lists case (leading coefficients of 1.96 and 1.32, respectively), a logical impossibility. Furthermore, spot checks done on a seventeen-word sentence show a leading coefficient of the same order of magnitude in the quadratic approximation (55.2 with candidates lists), but an order of magnitude smaller in the cubic approximation (0.27).

On the assumption that the divisions lists grow as n^3, least squares analysis shows the leading terms to be

Without Candidates Lists: $11.70\ n^3$

With Candidates Lists: $9.49\ n^3$

Again the quartic approximation for candidates lists is asymptotically greater than for non-candidates lists (leading coefficients of .238 and .147 respectively), leading one to suspect the growth is in fact cubic.

Cubic and quartic approximations of the rate of growth of time requirements had the following leading terms:

	Cubic	Quartic
Without Candidates Lists:	.0309	.00109
With Candidates Lists:	.0226	.00107

No empirical evidence could be found for preferring one approximation over the other. We note that, theoretically, because of overhead in parsing with candidates lists, the time requirements could be greater than parsing without.

Of particular interest is the comparative behavior of the algorithms with and without candidates lists. The ratio of the number of spans generated with candidates lists to the number generated without are as follows:

Word:	1	2	3	4	5	6	7	8	9
Ratio:	.470	.455	.458	.473	.496	.523	.546	.567	.587

(3)

The ratios of the number of divisions lists generated with candidates list to the number without are as follows:

Word:	1	2	3	4	5	6	7	8	9
Ratio:	.424	.404	.406	.424	.450	.480	.507	.534	.558

In both lists we note a dip near the beginning, indicating the fairly strong constraints the first few words of a sentence impose on the possible interpretations. Too many spans are generated however, resulting in a very gradual loss in effectiveness in the device of candidates lists.

The corresponding ratios for time requirements are

Word:	1	2	3	4	5	6	7	8	9
Ratio:	.667	.601	.571	.562	.567	.578	.593	.608	.623

Here we notice a similar dip at first followed by a gradual rise. That the percentages here are higher reflects the overhead in using candidates lists, and the particularly high ratio for the first word reflects the cost of initialization.

It is traditional to present figures on the number

of parses produced for the sentences, but here that is not terribly pertinent. The number of parses necessarily grows exponentially, but to say this is to say nothing about the algorithm, which localizes the ambiguity.

It is of greater interest to note how many of the spans are relevant, i.e. take part in a final parse of the sentence. This can be determined simply by following the divisions lists down the list of spans from a root node spanning the entire sentence. This is of particular interest since we have no theoretical bounds to guide our intuition. Does the number of relevant spans keep pace with the n^2 rate of growth of the number of all spans, or does the earlier part of a sentence impose enough constraints on the possible interpretations of later parts to slow this rate down significantly, or are enough spans generated in the early parts of a sentence to permit almost anything to be relevant in the latter parts, thus speeding up the rate of growth of relevant spans?

The average number of relevant spans for all nine-word sentences were

Word:	1	2	3	4	5	6	7	8	9
	32.0	92.2	188.3	325.8	513.0	746.7	1005.7	1298.4	1594.4

Of course the number of relevant spans is the same for the algorithm with and without candidates lists. It would be disturbing indeed if the candidates lists were to eliminate valid parses.

The quadratic least squares approximation has a leading term of 18.8 n^2. Since the cubic approximation has a negative leading term, it seems likely that the growth is more nearly quadratic.

Also of interest is the ratio of the number of relevant spans to the total number of spans, both with and without candidates lists.

Word:	1	2	3	4	5	6	7	8	9
Without Candidates Lists:									
Ratio:	.299	.300	.311	.322	.332	.335	.330	.324	.313
With Candidates Lists:									
Ratio:	.636	.659	.678	.682	.668	.640	.605	.571	.534

Without candidates lists, a little over 30% of the spans generated turn out to be relevant. With candidates lists, the proportion is between 60% and 65% although there is a sharp drop near the end of the sentence, reflecting the degradation in effectiveness of candidates lists shown in (3) above. These figures are perhaps more significant than the figures of (3). There it is seen that candidates lists reduce the number of spans generated by approximately 40% to 55%. Here it is seen that the number of <u>non-relevant</u> spans are generally reduced by between 70% and 80%, and even after nine words, the number of non-relevant spans is reduced by nearly 60%.

Results for the Non-Merging Algorithm: For the non-merging algorithm we have no such clean theoretical results

bounding the number of divisions lists and the time requirements. The longest production in the grammar is ten symbols. Therefore the theoretical bound on the growth of time requirements and the number of divisions lists is $K\ n^{11}$, but we could hardly be expected to detect this in sentences of eight or nine words. Thus, the only questions we can ask are, "How fast do the space and time requirements grow in the early parts of a sentence? In particular, will they really get out of hand as rapidly as the theoretical bounds suggest?"

The averages over all eight-word sentences were as follows:

Word:	1	2	3	4	5	6	7	8
Number of Complete Spans: Without Candidates Lists:								
	37.7	120.1	232.4	400.7	609.4	869.6	1185.8	1541.6
With Candidates Lists:								
	20.0	64.1	127.9	228.2	358.3	525.6	735.8	976.7
Number of Divisions Lists: Without Candidates Lists:								
	38.1	146.9	341.8	737.9	1431.7	2725.8	5103.8	9127.6
With Candidates Lists:								
	15.7	60.7	152.3	360.2	779.2	1664.8	3463.7	6704.4
Time Requirements: Without Candidates Lists:								
	.077	.309	.749	1.592	3.025	5.520	9.829	16.900
With Candidates Lists:								
	.053	.195	.462	.981	1.909	3.627	6.802	12.284

The growth in the number of complete spans is theoretically quadratic. Least squares analysis gives the following leading coefficients:

Without Candidates Lists: $23.3\ n^2$

With Candidates Lists: $16.3\ n^2$

Again the cubic approximation gives an asymptotically greater rate of growth for the candidates lists than for the non-candidates lists (leading coefficients of .497 and .344 respectively); this indicates that the rate of growth is in fact quadratic.

The leading coefficients of the cubic and quartic approximations for number of divisions lists and time requirements both without and with candidates lists are as follows:

Number of Divisions Lists:	Cubic	Quartic
Without Candidates Lists:	41.2	5.08
With Candidates Lists:	36.4	4.86
Time Requirements:		
Without Candidates Lists:	.067	.0079
With Candidates Lists:	.056	.0073

Tests were made of an exponential growth hypothesis as well, with similarly inconclusive results. The only conclusion to be drawn is that the algorithm is so disastrously bad that not enough data can be generated to draw any conclusions about how bad it is.

The following ratios indicate the comparative performance of the nonmerging algorithm without and with candidates lists:

Word:	1	2	3	4	5	6	7	8
Ratio of Number of Complete Spans With to Without Candidates Lists:								
Ratio:	.530	.534	.550	.570	.588	.604	.621	.634
Ratio of Number of Divisions Lists With to Without Candidates Lists:								
Ratio:	.412	.413	.446	.488	.544	.611	.679	.735
Ratio of Time Requirements With to Without Candidates Lists:								
Ratio:	.688	.631	.617	.616	.631	.657	.692	.727

As with the merging algorithm, there is a degradation in the effectiveness of the device of candidates lists with longer sentences, although this degradation is somewhat more pronounced in the nonmerging case. At first the ratio of time requirements improves somewhat as the parsing processor recovers from the initialization costs of candidates lists, but thereafter degradation also sets in.

The typical growth rate for the number of relevant spans is indicated by the averages for eight word sentences:

Word:	1	2	3	4	5	6	7	8
	15.0	46.7	91.3	157.9	235.6	323.5	439.2	584.4

The quadratic least squares approximation of this has a leading term of $8.51\ n^2$.

The ratios of the number of relevant spans to the total number of spans generated are

Word:	1	2	3	4	5	6	7	8
Without Candidates Lists:								
Ratio:	.398	.389	.393	.394	.387	.372	.370	.379
With Candidates Lists:								
Ratio:	.750	.729	.714	.692	.658	.615	.597	.598

These figures are slightly better than with the merging algorithm, although this certainly doesn't make up for the merging algorithm's advantages in other regards. Here also there is a perceptible decline in the effectiveness of candidates lists for longer sentences.

Because of their different data structures, the only direct comparison that can be made of the merging and nonmerging algorithms is on the basis of time. The following are the ratios of time requirements of the nonmerging algorithm to the merging algorithm:

Word:	1	2	3	4	5	6	7	8
Without Candidates Lists:								
	0.987	1.037	1.008	1.043	1.073	1.141	1.269	1.426
With Candidates Lists:								
	1.019	1.089	1.090	1.143	1.195	1.296	1.481	1.706

In other words, the nonmerging algorithm with candidates lists requires about 70% more time than the merging algorithm for eight word sentences and its relative performance only becomes worse for longer sentences.

Conclusions: The nonmerging algorithm is out of the question. The merging algorithm is better, but it is still necessary to apply the restrictions during parsing if we wish to parse any more than the shortest sentences.

The candidates lists is a powerful tool, which reduces space and time requirements by approximately 50% to 60% and reduces the number of irrelevant spans by close to 70%.

These conclusions influence the basic structure of the restriction metalanguage built up in the next chapter.

CHAPTER 5. A METALANGUAGE FOR EXPRESSING GRAMMATICAL RESTRICTIONS.

5.1. Introduction and Terminology.

The impracticality of nodal spans parsing in the absence of restrictions is convincingly demonstrated in Chapter 4. The problem of achieving nodal spans parsing is therefore the problem of expressing the restrictions in a form appropriate to the algorithm. In particular, the restrictions must be expressed in such a way that they act strictly locally. Because different interpretations of an ambiguous portion of a sentence may be merged under one span, it is not possible to traverse the parse tree easily as it is in top-down parsing. The information needed by a restriction must be anticipated and made readily available at the point at which the restriction is applied. Chapter 5 describes and defines a language developed for this purpose. Chapter 6 and Appendix B illustrate its use in expressing the restrictions of the Sager grammar.

We will assume the parser uses the merging form of the nodal spans algorithm given in Section 4.4. That is to say, the algorithm treats the context-free grammar internally as a binary grammar by leftward grouping. Certain terminology that will be useful in discussing the operation of the algorithm is best defined by means of an example. Suppose we have the production

B → D E F G

Suppose that the symbol D derives a substring of the input string. A span will be generated indicating this fact; this span will be called a span for D. Suppose that the symbol E derives a successive substring of the input string. The algorithm seeing the span for D and the span for E will generate a new span indicating that the first two symbols in the production for B have been recognized. The span for D will be called the first span, the span for E the second span, and the new span that is formed the new span. It will be said that the first span and the second span have been concatenated to form the new span, and that the first two symbols in the production for B have been matched. If the first k symbols in a production for B have been matched, the span generated will be called a span for B_k. Thus the spans for D and E have been concatenated to form a span for B_2. Now suppose we have the production

A → B / C

and suppose the symbols B and C derive the same substring. The algorithm on seeing the span for B generates a span for A and on seeing the span for C adds to the divisions list of the span for A. In this case it will be said that the spans for B and C have been merged under the span for A.

We assume in the remainder of this work that the grammar is available to the parser in the inverse form described in Section 4.2. That is, for each symbol of the grammar there

is a set of entries, one for each occurrence of the symbol on the right side of a production. Each entry contains the location triple <A,k,j> of the occurrence, flags indicating whether the symbol is possibly initial, necessarily initial, and possibly terminal in that occurrence, and a list of the symbol's possible successors. In addition, each entry includes the entire set of restrictions acting at that point, i.e. the restrictions which must be applied before a new span dictated by that entry in the grammar is formed. The algorithm of Section 4.4 is modified in that an exit is made before a new span is formed to check the restrictions applying at that point.

It will be recalled from Section 2.5 that words are listed in the dictionary as tree structures. The entry for "faces", for example, is as in Figure 1. We will refer to this as the dictionary entry for the word "faces". The root node of the dictionary entry will be called its literal.

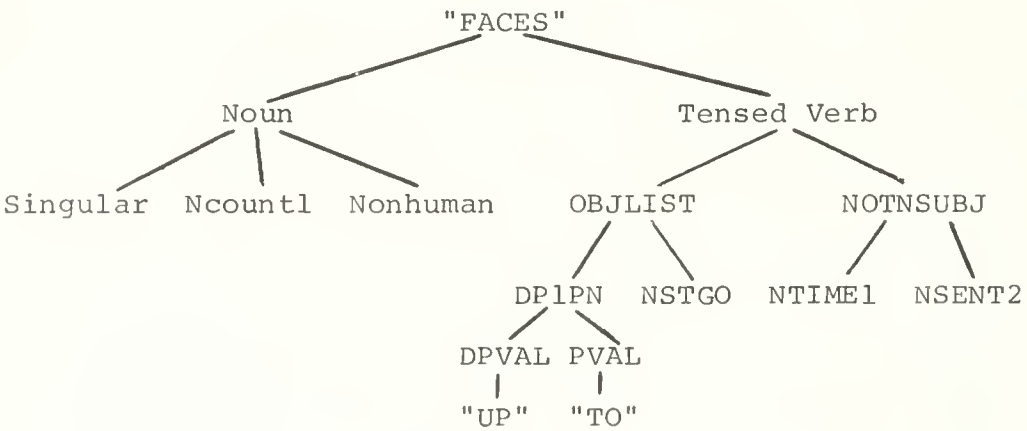


Figure 1

The immediate descendents of the literal will be called categories. Descendents of category nodes will be called attributes. A special kind of attribute is one which may have more than one immediate descendent node, as the OBJLIST and NOTNSUBJ attributes of Figure 1; such attributes will be called list attributes. Before the parsing begins, lexical scanner and dictionary lookup routines will make available to the parser the dictionary entry of every word in the sentence to be parsed. In its first step with each successive word, the parsing algorithm generates a separate span for every category in the word's dictionary entry. This category together with all its descendent attributes and its parent literal will be called a lexical item. Thus there will be a distinct lexical item for every category of each word in the sentence.

5.2. General Description of the Restriction Metalanguage; The Core.

The underlying motivation for the restriction metalanguage is a rather strong form of Sager's principle that there is no action at a distance in a sentence. [1] Grammatically related elements are adjacent at some point in the parse tree. Hence the restriction expressing this

[1] Naomi Sager, "A Hierarchy of BNF Grammars for Parsing English: A Sublanguage Approach," talk delivered at University of California, Berkeley, Department of Electrical Engineering and Computer Sciences, April 14, 1971.

grammatical relationship can be invoked when the spans for these adjacent elements are to be concatenated. Restrictions usually depend on the attributes of the "core words" of adjacent elements. Therefore, it ought to be sufficient to associate with each span a lexical item as its "core word" and express restrictions in terms of the literals, categories and attributes of these lexical items.

For example, in both the sentences

"Descartes thought."

"The brilliant Descartes thought deeply on fundamental questions."

when the subject and the verb phrase are to be concatenated to form an assertion, the "core word" of the subject will be the lexical item for "Descartes" and the "core word" of the verb phrase the lexical item for the tensed verb category of "thought." [2] These lexical items can be compared to see that they have compatible attributes, i.e. that "Descartes" is something that can think. If so, the restriction succeeds and the new span for the assertion is formed.

Unfortunately, the literal, category, and attributes of the lexical item defined as the "core word" do not appear to suffice for the expression of all the restrictions. As an example, consider the following two sentences:

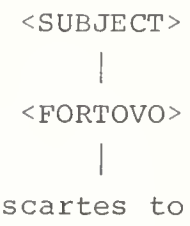
"For Descartes to think was for him to exist."

"For Descartes to think was to exist."

---

[2] "Thought," of course, is also a noun and a past participle.

In the first the analysis of the first four words is



In the second the analysis is

In a nodal spans parse using binary merging, both analyses are merged under the span for $\langle ASSERTION \rangle_2$. Yet when we reach the object of "be", one analysis is valid and the other must be rejected.

Two difficulties are seen in this example. First the merging of the two interpretations of the initial part of the sentence under the span for $\langle ASSERTION \rangle_2$ must be avoided or it will be necessary later to restructure the span collection to eliminate one of the interpretations. Second, when the restriction is invoked, information is required which is not contained in any lexical item in the sentence, i.e. information as to what kind of string is taken as the subject. Both difficulties can be overcome by including as part of the "core word" a number of flags. These flags can be set at one point in the parsing to signal

a particular structural feature, passed up to the level at which this information is required, and then unset.

More formally, a flag is a string of alphanumeric characters ending in "flg". The core of a span consists of a lexical item and a set of flags. We may speak of the lexical item of a core as the core lexical item, and of the literal, category or attributes of the core lexical item as the literal, category or attributes of the core. When a first span and a second span are concatenated to form a new span, the core of the new span is formed at the same time. The lexical item of the new core can be the lexical item of the core of either the first span or the second span. The inverse grammar entry which generates the new span indicates whether the core lexical item of the first span or the second span is to be taken as new core lexical item. The set of flags of the new core is constructed in accordance with explicit instructions in the restrictions applying at that point. When a flag is added to the set of flags of the new core, we say that the flag is set. When it is removed from the set of flags, we say it is unset. We say a flag is passed up to a new core from the core of the first or second span when it is set if and only if the core of the first or second span, respectively, contains that flag. The restriction component consists of instructions which set, pass, and unset flags, and statements which check the flags, literals, categories and

attributes of the cores of the spans being concatenated.

The core of a span is carried along with the span as one of its determiners -- if $core_1$ and $core_2$ differ, then the spans <p,A,q,$core_1$> and <p,A,q,$core_2$> differ and no merging takes place. Two cores are the same if and only if they contain the same flags and have the same lexical item.

In the above example, when forming <SUBJECT> from <FORTOVO>, the FORTOVOLOFLG [3] is set. When forming <SUBJECT> from <TOVO>, the TOVOFLG is set. The FORTOVOFLG and TOVOFLG are passed up to the level of $<ASSERTION>_2$, giving us two distinct spans for $<ASSERTION>_2$ over the same substring, and passed along the <ASSERTION> string until <OBJECT> is encountered. There the restriction is applied, one of the interpretations is rejected, and the flags unset as they are no longer needed.

When a string is being matched, the span for the string will take as its core lexical item the core lexical item of one of its elements. This element will be called the core element of the string. The core elements of strings are defined explicitly in the BNF component of the grammar (see Appendix A where the core element of a string is indicated

---

[3] The names of the particular flags, of course, are arbitrary. Here we will use the names of flags actually used in the application of the metalanguage to the Sager grammar in Chapter 6. To an extent, they are mnemonic. A complete alphabetical list of the flags used for the Sager grammar and their intended significance is given in Appendix C.

by a "C" following that element). This explicit definition is necessary since the element to be chosen as the core element of a string is determined by the restrictions which act on constructions involving that string and no dependable regularities are apparent. The following general remarks may be made, however: The core of a span for a category symbol is the lexical item which generates the span. In the singleton production "A → B" the core lexical item of the span for B becomes the core lexical item of the span for A. The <LXR> type symbols, i.e. those expanding into a variant flanked by left and right adjuncts --

<LXR> → [<LX>] <XVAR> [<RX>]

-- have the variant <XVAR> as the core element rather than the adjuncts. <ASSERTION> and the other center strings, sentence nominalizations, and verb-object strings generally have the verb as the core element, although this is sometimes overridden by explicit instructions in the restriction component, as explained in Section 6.2 below. Prepositional phrases and other constructions involving prepositions have the preposition as the core element. Wh-strings involving the element <ASSERTION> have <ASSERTION> as the core element. The core element of the left adjunct of the noun is explained in Section 6.3 below. Adjuncts are never taken as the core element of a string.

As an example, consider the sentence

"The brilliant Descartes thought deeply on fundamental questions."

The noun phrase "fundamental questions" has the noun "questions" as its core lexical item. When it is concatenated with "on", the preposition becomes the core lexical item of the prepositional phrase. The tensed verb "thought" is the core lexical item of the span for <LTVR> subsuming "thought deeply" and also of the entire verb phrase. "Descartes" is the core lexical item of the noun phrase "The brilliant Descartes" and when this is concatenated with the verb phrase the tensed verb "thought" is the core lexical item of the entire assertion.

5.3. Housing Specifications

The restrictions are invoked upon application of an entry in the inverse grammar and executed before the new span is formed, and so are housed at these entries. For example, a restriction housed at D in the production

$$A \rightarrow B\ C\ D\ E$$

is executed before a span for D is concatenated with a span for A_2 to form a span for A_3. Each restriction therefore begins with a listing of the entries at which it is housed. The basic listing is of the form

HOUSED AT <SYMBOL>: <PRODHD>,k,j:

This means that the restriction is housed at the occurrence of the symbol <SYMBOL> as the jth element in the kth alternative in the production for <PRODHD>. If the restriction is to be applied at several occurrences of <SYMBOL> in the grammar, this may be indicated by the listing

HOUSED AT <SYMBOL>: <PRODHD1>,k_1,j_1; <PRODHD2>,k_2,j_2,...;
<PRODHD3>,k_3,j_3:

The location triples "<PRODHDi>,k_i,j_i" are separated by semicolons and the entire housing specification ends with a colon. If the restriction is to be applied at some occurrences of several symbols in the grammar, the listing becomes

HOUSED AT <SYMBOL1>: <PRODHD1>,k_1,j_1; ...; <PRODHD2>,k_2,j_2;
<SYMBOL2>: <PRODHD3>,k_3,j_3; ...; <PRODHD4>,k_4,j_4;
... ;
<SYMBOL3>: <PRODHD5>,k_5,j_5; ...; <PRODHD6>,k_6,j_6:

The specifications of occurrences for each of the symbols are separated by semicolons and the entire housing specification ends with a colon.

The alternative number, k, may be omitted if <SYMBOL> occurs in no more than one alternative of the production for <PRODHD>. If <SYMBOL> occurs only once in the entire production for <PRODHD>, both the alternative number, k, and the element number, j, may be omitted. Thus

HOUSED AT <SYMBOL>: <PRODHD1>,j_1; <PRODHD2>:

is also a valid housing specification. For example, the housing of a restriction at D in the production

A → B C D E / F G

may be indicated as

HOUSED AT D: A, 1, 3:

or by

HOUSED AT D: A, 3:

or by

HOUSED AT D: A:

Another valid housing specification is

HOUSED AT <SYMBOL>: ALL OCCURRENCES:

which causes the restriction to be housed at all occurrences of <SYMBOL> in the inverse grammar. The specification

HOUSED AT <SYMBOL>: ALL OCCURRENCES EXCEPT <PRODHD1>,k_1,j_1; <PRODHD2>,j_2; <PRODHD3>:

causes the restriction to housed at all occurrences of <SYMBOL> except the listed ones.

Housing specifications can be condensed by the following device: If a restriction is to be invoked both at symbol A and symbol B in the production for C, this may be represented

HOUSED AT A, B: C:

Extending this, if a restriction is to be invoked at the occurrences of A and B in the productions for C, D, and E, it is represented

HOUSED AT A, B: C; D; E:

When the grammar is compiled, the restrictions are stored with the inverse grammar entry at which they apply.

5.4. Flag- and Attribute-Checking Statements

The basic checking statement is of the form

CORE1 HAS(NT) eltstring.

or

CORE2 HAS(NT) eltstring.

When two spans are to be concatenated to form a new span, CORE1 always refers to the core of the first span and CORE2 always refers to the core of the second span. Thus, in the production

A → B C D E

when concatenating a span for D with a span for A_2 , the core of A_2 is called CORE1 and the core of D is called CORE2. The term CORE2 is used when a span is to be matched with the head of a string or with a singleton production. In such cases we may say there is no first span and the set of flags of CORE1 is assumed to be empty. This may seem non-intuitive, but allows for regularity in the expression of the restrictions.

"eltstring" in the basic checking statement consists of flags (like TOVOFLG and FORTOVOFLG), categories and attributes (like TV or PLURAL or NTIME1), and literals (like "IT" or "WHEN"), joined by the logical connectives AND, OR and NOT.

The statement

COREx HAS elt.

returns the value "true" if COREx contains the indicated flag, category, attribute, or literal, "false" otherwise. "COREx HAS eltstg1 AND eltstg2" is interpreted as "COREx HAS eltstg1 AND COREx HAS eltstg2." "COREx HAS eltstg1 OR eltstg2." is interpreted as "COREx HAS eltstg1 OR COREx HAS eltstg2." "COREx HAS NOT eltstg" and "COREx HASNT eltstg" are interpreted as "NOT (COREx HAS eltstg)."

List-attributes like OBJLIST and NOTNSUBJ may be treated as functions taking cores as arguments and yielding sets of attributes, flags, or literals as values. To facilitate use of such functions, two more checking statements

are included in the language -- set-membership and empty-intersection checking. The basic set-membership checking statement is typically of the form

elt (N)EL listattribute (COREx) .

where "elt" is a flag, attribute, or literal, and "list-attribute" is a list-attribute. The value of "listattribute(COREx)" is the set containing the immediate descendent attributes, flags, or literals of the list-attribute. "elt EL listattribute(COREx)" returns a value "ture" if the flag, attribute, or literal is a member of this set, "false" otherwise. "elt NEL listattribute(COREx)." is interpreted as "NOT (elt EL function(COREx))." "elt" may also be "CORE1" or "CORE2" in which case "listattribute (COREx)" is searched for the literal of CORE1 or CORE2 respectively.

The empty-intersection checking statement

tokset1 * tokset2 = NL.

compares two sets of tokens and returns "true" if they have no attributes, flags, or literals in common, "false" otherwise. The token sets can be "listattribute(COREx)" as explained above, or CORE1 or CORE2, in which case the included tokens are the literal, the category, and the top-level attributes of the core lexical item. "tokset1 * tokset2 ≠ NL." is interpreted as "NOT (tokset1 * tokset2 = NL)." A typical use of the empty-intersection checking statement is the restriction

CORE1 * NOTNSUBJ(CORE2) = NL.

called when <VERB> is to be concatenated with <SUBJECT>, which checks the attributes of the subject for ones which cannot cooccur with the verb.

Basic checking statements may be joined by means of logical connectives --

IF checkingstatement1 THEN checkingstatement2.

checkingstatement1 AND checkingstatement2.

checkingstatement1 OR checkingstatement2.

NOT checkingstatement.

and nested by means of parentheses --

checkst1 AND (checkst2 OR checkst3).

5.5. Forming the New Core; Flag-Setting Instructions.

Normally when a new span is formed, the span inherits a core lexical item from either CORE1 or CORE2. The entry in the inverse grammar indicates which constituent's core lexical item is to be taken as the new core lexical item. In some circumstances, the choice of core element depends upon certain conditions expressed by checking statements. Then a restriction of the form

IF checkingstatement THEN CORE HAS ATTRIBUTES OF CORE1
ELSE CORE HAS ATTRIBUTES OF CORE2.

is invoked. The word CORE always refers to the core of the new span being formed.

Flags may be set in the new core by means of the instruction

CORE has flag.

Several flags may be set at once by using the instruction

CORE HAS flag1, flag2, flag3.

Flags are not inherited automatically by the new core as are the lexical items. There are two cases. If the new span is an extension along a string production of an incomplete span, the new core automatically inherits the flags of the old core unless blocked by an instruction

CORE HASNT flag1, flag2, ... (1)

If the new span is a new production head, then none of the flags are passed up from the old cores except those explicitly stated in instructions

CORE HAS flag1, flag2, ... OF CORE1.

CORE HAS flag1, flag2, ... OF CORE2.

For example, in the grammar fragment

A → B

B → C D E F ,

suppose a flag is set when D is concatenated with C to form a span for B_2. This flag is automatically retained in the cores of the spans for B_3 and B_4 (or B), unless otherwise stated, but unless there is an explicit instruction passing it up, it is dropped when the new span for A is formed.

The latter feature is included because it is more natural for the restriction writer to specify where a flag is to be active by setting and passing it than to specify where it is to be inactive by unsetting it. The former

feature, automatic passing along a string, is adopted because although the parser acts as though the grammar were binary, it should not be necessary for the restriction writer to keep that in mind.

This automatic passing of flags may result in a flag being active slightly longer than it is actually needed and thus preventing merging that might otherwise occur. This would have no effect on the parses ultimately produced, and it is not expected that it would increase significantly the number of spans generated. If it turns out to cause difficulties one may introduce a flag-unsetting instruction (1).

An instruction which is not necessary but may result in fewer spans being generated is

CORE HASNT ATTRIBUTES.

which eliminates reference to the core lexical item, so that the core consists only of a set of flags. This may be invoked at a point high enough in the span collection that reference to lexical items in the sentence is no longer needed to execute the restrictions.

Flag setting instructions may be made conditional by the formats

IF checkingstatement THEN settinginstrl ELSE settinginstr2.

IF checkingstatement THEN settinginstr.

5.6. Endpoint-Checking Statements; Comma Restrictions.

At the present time in the Sager grammar "." (except in abbreviations), "?", and ";" are viewed as endmarks and counted as a word in the sentence. Material between parentheses is ignored, and there are no provisions for colons or dashes in the grammar. Thus commas are the only punctuation marks which play a significant role in the parsing of a sentence. Commas are not counted as words but rather assigned to inter-word points. We say a comma is assigned to the ith interword point if there is a comma after the ith word in the sentence. The commas are kept in an array called INTER. There they can be referred to in comma restrictions by means of statements involving the word INTER, e.g.

INTER(i) = ",". INTER(i) ≠ ",".

Otherwise the commas are ignored.

If other punctuation marks are ever incorporated into the grammar they can be handled in a similar way.

Another operator which may be called is "WORD", as in

WORD(i) ≠ "THE".

meaning the ith word is not "THE".

In the current implementation the argument i of both INTER and WORD may take on only the values P, P+1, Q, Q+1. P stands for the initial point and Q the terminal point of the second span (the span corresponding to CORE2). The values of i are so constrained in order to keep the restrictions independent of the length of the sentence, but

this constraint could easily be lifted.

In the application to the Sager grammar, the operator INTER is used only in comma restrictions. The operator WORD is used rarely.

Another kind of endpoint-checking statement is of the form

P = integer. Q = integer.

P ≠ integer. Q ≠ integer.

The only use made of this currently is in one restriction which checks if P = 0, i.e. if the node spans an initial portion of the sentence.

These checking statements may be combined by logical operators in the same way as the flag- and attribute-checking statements outlined above.

5.7. Span Checking Statements.

Recall that when a span is to be matched with the head of a production or with a singleton production, it is considered the second span (corresponding to CORE2). The first span is null. The checking statement

SPAN1 IS NL.

returns "true" if this is the case, "false" otherwise. The checking statement

SPAN1 ISNT NL.

is interpreted as NOT (SPAN1 IS NL). These statements may be combined with other checking statements by logical operators in the usual way.

These statements prove useful in restrictions governing the omission of nouns in wh-strings.

5.8. Semigrammaticality Markers.

There are two sorts of restrictions expressible in the metalanguage: those involved in the formation of the new core and those checking conditions to see if a new span may be formed at all. In Section 2.3 a program was envisioned of successively lifting classes of checking restrictions until a parse is obtained on some level of a predefined hierarchy of semigrammaticality. In a nodal spans environment it is not necessary to perform successive parses. If the parser is operating in semigrammaticality mode and a checking restriction returns "false", instead of the new span being rejected, it is constructed and tagged with a semigrammaticality marker corresponding to the class of the restriction.

When the parsing is completed, the valid parses are read off the list of spans with all the restrictions in effect. If no parse is found the maximal well-formed substrings are output, one class of restrictions is lifted, and the list of spans is again read off, this time for a parse on a lower level of grammaticality. With each successive reading off, another class of restrictions in the hierarchy is lifted until a parse is found on some level of semigrammaticality.

A semigrammaticality marker is indicated by a dollar

sign followed by a string of alphanumeric characters, e.g. $PUNC. The place of a checking restriction in the semi-grammaticality hierarchy is indicated by appending the marker to the front of the restriction with OR:

sgtag OR checkingrestriction.

The semigrammaticality markers, descriptions of their corresponding classes of restrictions, and a recommended hierarchy for the Sager grammar, are given in Chapter 7.

5.9. Syntax of the Restriction Metalanguage

The complete syntax of the restriction metalanguage is given below. The specification is in standard BNF notation: The triangular brackets enclose symbols of the BNF description language. Symbols beginning with "<*..." are atomic. "→" stands for "derives". "/" separates alternatives. Square brackets "[...]" indicate that what is enclosed is optional. All other symbols are part of the restriction metalanguage.

The atomic symbols of the grammar are as follows:

<*name>,<*attribute>: string of alphanumeric characters beginning with an alphabetic character and not ending in "FLG".

<*intsym>: string of alphanumeric characters enclosed in triangular brackets.

<*atom>: "<" followed by "*" followed by a string of alphanumeric characters followed by ">".

<*literal>: string of alphanumeric characters enclosed in quotes.

<*integer>: an integer.

<*sgtag>: "$" followed by a string of alphanumeric characters.

<*flag>: string of alphanumeric characters ending in "FLG".

```
<component>   → <restriction> [<component>]
<restriction> → <*name>: <bulk>.
<bulk>        → <house>:<statements> [.<bulk>]
<house>       →  HOUSED AT <locs>
<locs>        → <symbols>: <refs> [;<locs>]
<symbols>     → <symbol> [, <symbols>]
<symbol>      → <*intsym> / <*atom> / <*literal>
<refs>        → <refstg> / ALL OCCURRENCES [EXCEPT <refstg>]
<refstg>      → <*intsym> [[, <*integer>],
                      <*integer>] [;<refstg>]
<statements>  → <statement> [. <statements>]
<statement>   → [<*sgtag> OR] <cond1> / IF <cond1>
                      THEN <set1> [ELSE <set1>] / <set1>
<cond1>       → IF <cond1> THEN <cond1> / <cond2>
                      [AND <cond1>] / <cond2> [OR <cond1>]
<cond2>       → <cond3> / (<cond1>)  / NOT <cond2>
<set1>        → <set2> [AND <set1>]
<cond3>       → <hasst> / <setmem> / <intersect> / <inter> /
                     <endptck> / <wordck> / <spanchek>
```

```
<hasst>          → <corex> <hasn> <bitstgl>

<corex>          → CORE1 / CORE2

<hasn>           → HAS / HASNT

<bitstgl>        → <bitstg2> [AND <bitstgl>] / <bitstg2>
                          [OR <bitstgl>]

<bitstg2>        → <bit> / (<bitstgl>) / NOT <bitstg2>

<bit>            → <*flag> / <attrstg> / <*literal>

<attrstg>        → <*attribute> [: <attrstg>]

<set2>           → CORE <hasn><flags> / CORE HAS <flags>
                          OF <corex> / CORE HAS ATTRIBUTES
                          OF <corex> / CORE HASNT ATTRIBUTES

<flags>          → <*flag> [, <flags>]

<setmem>         → <elt><eps><fcore>

<elt>            → <*flag> / <*attribute> / <*literal> /<corex>

<eps>            → EL / NEL

<fcore>          → <corex> / <attrstg> (<corex>)

<intersection>→ <fcore> * <fcore> <eq> NL

<eq>             → = / ≠

<inter>          → INTER (<pq>) <eq><*literal>

<pq>             → P / P+1 / Q / Q+1

<endptck>        → <pq><eq><*integer>

<wordck>         → WORD (<pq>)<eq><*literal>

<spanchek>       → SPAN1 <isn> NL

<isn>            → IS / ISNT
```

5.10. Semantics of the Metalanguage.

The semantics of the housing specifications should be clear from the description in Section 5.3.

The semantics of the body of a restriction is given in this section. If the substitutions listed below are carried out in the order indicated, the restriction will be transformed into a SETL expression or instruction based on functions coded below. The symbols in triangular brackets are those of the previous section.

Initial Substitutions:

<*name> : <bulk>. → definef <*name>; return <bulk>;
end <*name>;

<house> : <statements>[.<bulk>] → <statements>[and <bulk>]

<statement>.<statements> → <statement> and<statements>

<*sgtag> OR <condl> is explained in Section 7.2.

IF <condl> THEN <condl> → (<condl>) imp (<condl>)

Distributive Transformations:

<corex> HASNT <bitstgl> → NOT(<corex> HAS <bitstgl>)

<corex> HAS <bitstg2> $\left\{\begin{matrix}\text{AND}\\ \text{OR}\end{matrix}\right\}$ <bitstgl> →

(<corex> HAS <bitstg2> $\left\{\begin{matrix}\text{AND}\\ \text{OR}\end{matrix}\right\}$ <corex> HAS <bitstgl>)

<corex> HAS NOT <bitstg2> → NOT <corex> HAS <bitstg2>

CORE <hasn><*flag>,<flags> → CORE <hasn><*flag>
and CORE <hasn><flags>

CORE HAS <*flag>, <flags> OF <corex> →
CORE HAS <*flag> OF <corex> and CORE HAS <flags>OF<corex>

<*attribute>: <attrstg> → <<*attribute>> + <attrstg>

<elt> NEL <fcore> → NOT (<elt> EL <fcore>)

SPAN1 ISNT NL → NOT SPAN1 IS NL

EL Substitution:

{CORE1 / CORE2} EL ... → lit(core({span1 / span2}) EL ...

{<*flag> / <*attribute> / <*literal>} EL <attrstg>(<corex>) →

<corex> HAS <attrstg> + <{<*flag> / <*attribute> / <*literal>}>

{<*flag> / <*attribute> / <*literal >} EL <corex> → <corex> HAS {<*flag> / <*attribute> / <*literal>}

Function Substitution:

{CORE1 / CORE2} HAS {<*flag> / < attrstg> / <*literal>} → check ({1 / 2}, {<*flag> ,1 / < attrstg>,2 / <*literal>,3})

CORE HAS <*flag> → set (<*flag>)

CORE HASNT <*flag> → unset (<*flag>)

CORE HAS <*flag> OF {CORE1 / CORE2} → pass (<*flag>,{1 / 2})

CORE HAS ATTRIBUTES OF {CORE1 / CORE2} → coreset ({1 / 2})

CORE HASNT ATTRIBUTES → coreunset

SPAN1 IS NL → isnl

<attrstg>(CORE1) → <attrstg>
CORE1 → nl
} call this <fcore1>

<attrstg>(CORE2) → <attrstg>
CORE2 → nl
} call this <fcore2>

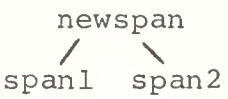
<fcore1> * <fcore2>
<fcore2> * <fcore1>
→ intersect (<fcore1>,<fcore2>)

P → pee(span2) Q → que(span2)

= → eq ≠ → ne

WORD(...) → literal(sentence(...))

Externals: When the restriction is called, there are three spans involved. "span1" and "span2" are being concatenated to form "newspan".



Each span is a 4-tuple, consisting of an initial point pee, and endpoint que, a symbol eta, and a core core.

<pee,eta,que,core>

The functions pee, eta, que, and core extract these four elements.

Each core is a triple, consisting of a set of flags, a category node of a word in the sentence, and a string which is the literal spelling of the word.

<{'aflg','bflg',...},catnode,'"man"'>

These three elements are extracted by the functions flags, catnode, and lit, respectively. The attributes of the core word are accessible by using the function desc on the category node. The names of the category and attributes are accessible using the function name.

The subroutine restrictiontest: restrictiontest is called with the two spans being concatenated, span1 and

span2, as arguments together with the entry occ in the inverse grammar that dictates this concatenation. First the flags of the new core are set from the core of span1 if it is not null, and the attributes and literal are determined from an indicator in the inverse grammar entry occ extracted by the function corbit. corbit(occ) returns t if the core of span2 is to be taken as the new core, f if the core of span1. Then all the restrictions stored at occ are extracted with the function restrlist and executed. All other externals in the algorithm have been described above.

```
define  restrictiontest(pas,newcore,span1,span2,occ);
/* the following are assumed to be global:  catnode, core,
     lit, flags, corbit, eta, divlis, restrlist */
/* form the new core */
<newflags,newcatnode,newlit> = if span1 eq nl
     then <nl, catnode(core(span2) is kor), lit(kor)>
     else <flags(core(span1) is korl), if corbit(occ) then
          <catnode(core(span2) is kor2), lit(kor2)>
           else <catnode(korl), lit(korl)>>; pas = t;
/* execute all the restrictions housed at that point
     in the grammar until one fails */
(∀restr ∈ restrlist(occ) while pas) pas = pas and restr; end ∀;
newcore = <newflags,newcatnode,newlit>;  return;
end restrictiontest;
```

The Functions Defined: The function check executes the checking statements and returns a value of t or f

according as the statement passes or fails.

```
definef check(x,elt,type);
if x eq 1 then kor = core(span1); else kor = core(span2);;
if type eq 1 then return elt ∈ flags(kor);
else if type eq 2 then <nod,elt> = elt; x = catnode(kor);
     flow          (name(x) eq nod)?
               reset,             destest?
                              (elt)?   return f;
                        reset,  return t;
     destest:  y=x; = (∃x ∈ desc(y)|name(x) eq nod);
     reset: <nod,elt> = elt;  to destest;
     end flow;
else return elt eq lit(kor);;
end check;
```

The function set executes the setting instructions by setting the new flags and returning t.

```
definef set(flag);
newflags = newflags with flag; return t;  end set;
```

The function unset is similar.

```
definef unset(flag);
newflags = newflags less flag;  return t;  end unset;
```

The function pass sets flags in the new core that are contained in the specified old core.

```
definef pass(flag,x);
if flag ∈ flags(core(if x eq 1 then span1 else span2))
then newflags = newflags with flag;;return t; end pass;
```

The function coreset sets the new core to either CORE1 or CORE2.

```
definef coreset(x);
kor = core(if x eq 1 then span1 else span2);
<newcatnode,newlit> = <catnode(kor),lit(kor)>;
return t;  end coreset;
```

The function coreunset unsets the core lexical item set previously in restrictiontest.

```
definef coreunset;
<newcatnode,newlit> = <nl,nl>; return t; end coreunset;
```

The function intersect executes the empty intersection checking statements.

```
definef intersect(elt1,elt2);
(1 ≤ ∀i ≤ 2) elt = if i eq 1 then elt1 else elt2;
       kor = core(if i eq 1 then span1 else span2);
flow                    (elt)?
         setup?            (s(i)={lit(kor)}; continue ∀;),
    reset,    destest?
          (elt)?   (s(i)=nl; continue ∀;),
       reset,  (s(i)=desc(x); continue ∀;);
```

[continued]

```
setup:    <nod,elt> = elt; x = catnode(kor); = name(x) eq nod);
destest:  y=x; = (∃x ∈ desc(y)|name(x) eq nod);
reset:    <nod,elt> = elt; to destest;
end flow;  end ∀;
return s(1) * s(2);          end intersect;
```

The function isnl determines if spanl is null.

```
definef isnl;
return spanl eq nl;  end isnl;
```

INTER is an array in the main program, most of whose entries are blank, but which has a "," in elements corresponding to inter-word points having a comma.

5.15. The Language is Still Context-Free.

The restriction metalanguage is purposely limited in power. A language recognized by a grammar consisting of BNF productions augmented by restrictions expressed in the restriction metalanguage is still a context-free language. That is, there is a context-free grammar which recognizes it.

To show this we will indicate how to construct a weakly equivalent purely context-free grammar G' out of a context-free grammar G augmented by restrictions. If <A> is a nonterminal or terminal symbol of G, then the corresponding symbols of G' will be

$$\{\langle A, \{s_i\ attr_i\}, \{s_j\ flg_j\}, \{s_k\ com_k\}\rangle\}$$

where the $attr_i$ include all possible attributes or all

possible branches of attribute trees of categories that the core word of A could have and all literal words mentioned in the restrictions consistent with those categories. flg_j ranges over all flags active at that node. com_k takes on the values "preceding comma" and "succeeding comma". s_i , s_j , s_k may be either + or -, depending on whether the attribute or comma is present or the flag is set at that point. If there are x possible core word attributes and y flags active at a node then corresponding to the one symbol of G there will be 2^{x+y+2} symbols in G'.

A production of G together with the restrictions housed at that point can be translated into productions of G' in a reasonably straightforward manner. Let A → B C be a production of G, and let $A'_1,\ldots,A'_\ell,B'_1,\ldots,B'_m, C'_1,\ldots,C'_n$, be the symbols of G' corresponding to A, B and C respectively. Begin by writing down the productions

$$A'_i \rightarrow B'_j\ C'_k\ , \quad i = 1,\ldots,\ell, \quad j = 1,\ldots,m\ , \quad k = 1,\ldots,n.$$

For each restriction of G housed in A → B C , and for each production $A'_i \rightarrow B'_j\ C'_k$, substitute the values indicated by A'_i , B'_j , C'_k into the restriction, evaluate it as a Boolean expression, and discard the production from G' if the resulting value is FALSE. For example, if the restriction is

IF CORE1 HAS BFLG AND CORE2 HAS CFLG THEN CORE HAS AFLG.

and if A'_i contains -AFLG, B'_j contains +BLFG, and C'_k contains -CFLG, then substitution results in the Boolean expression

IF (TRUE AND FALSE) THEN FALSE

whose value is TRUE. Thus the production remains in G'.

All flag-aand attribute-checking and flag-setting restrictions can be handled in this manner. Flag-setting instructions involving

IF ... THEN ... ELSE ...

may first be translated into

IF ... THEN ...

IF NOT ... THEN ...

Flag-passing instructions

CORE HAS AFLG OF CORE2.

may be translated into

(CORE HAS AFLG AND CORE2 HAS AFLG) OR (CORE HASNT AFLG AND CORE2 HASNT AFLG).

Set-membership and empty-intersection checking restrictions can be translated into flag- and attribute-checking restrictions and therefore may also be handled in this way.

A restriction

INTER(P) = ",".

rejects a production in which C'_j contains "-preceding comma" and other INTER restrictions are handled similarly. WORD and endpoint-checking restrictions occur rarely and are allowed as a convenience rather than as a necessity and could be eliminated from the grammar G itself.

Of course, this procedure is not recommended for real grammars because of the enormous increase in the size of the grammar. In fact, this increase points to the power of

the restriction component of the grammar. If the present grammar which consists of some 200 BNF productions and some 300 restrictions were translated into a pure context-free grammar in this manner, the number of productions in the resulting grammar would be several orders of magnitude greater.

CHAPTER 6. THE METALANGUAGE APPLIED TO THE SAGER GRAMMAR FOR ENGLISH.

6.1. Introduction.

First of all, in order to demonstrate the usefulness and power of the restriction metalanguage, and secondly, in order to produce a working, fast, bottom-up parallel parser of English, the restrictions of the Sager grammar were written in the metalanguage. No attempt was made to deal with conjunctions or comparatives, but otherwise the entire restriction component was handled. This includes agreement restrictions between the subject and verb phrase and within the noun phrase, selectional restrictions between the subject and the deepest verb, restrictions governing the cooccurrence of deeply nested sentence nominalizations in the subject and the predicate, and restrictions governing the omission of nouns from deeply embedded wh-strings. No restriction was found which could not be expressed in the metalanguage, and in fact the same approach could be used for every restriction. Restrictions acting in the lower levels of the grammar, near the atomic level, could generally be expressed in terms of core lexical items. Many positional, quantifier, and noun phrase restrictions fell into this class. Restrictions acting at higher levels of the grammar, the sentence nominalization, agreement, and wh-string restrictions, could usually be expressed by

setting a flag at a lower level in the parse tree to indicate the presence of some structural feature, passing it up to the higher level, and executing the restriction on the basis of the flags present at that node. The restriction component coded in the metalanguage including an explanatory comment for each restriction is listed in its entirety in Appendix B. The BNF component as modified for nodal spans parsing is given in Appendix A along with a guide to the mnemonics and examples of the strings.

There are two results which this project has demonstrated. The ideal machine grammar for scientific English recognizes all grammatical sentences and only those. Work done at the String Project indicates that the Sager grammar comes impressively close to this ideal. Insofar as it does, the result of Section 5.11 shows that, theoretically, scientific English can be recognized by a purely context free grammar. Second, a method exists for parsing scientific English in time n^4. The extra power of n is introduced because in the span <p,A,q,core> the core lexical item can refer to any word between the (p+1)st and the qth. (Of course, if scientific English is a context-free language, it can be recognized in time $K\ n^3$, but what a K !) These remarks are of theoretical interest only; it is not claimed that a context-free grammar of English could be explanatory.

The next two sections of this chapter discuss modifications which the algorithm forces upon the grammar. This is

of linguistic interest. Sections 8.4 and 8.5 are apologies for restrictions which are handled inelegantly. In Section 8.6 several restrictions of increasing complexity are presented and it is shown how to code these restrictions in the metalanguage. In Section 8.7 all the restrictions acting at one point in the grammar are explained. These two sections should impart the flavor of both the restriction component of the Sager grammar and of the restriction metalanguage. The chapter closes with a SETL specification of the complete parsing algorithm with all the bells and whistles it has acquired along the way, and a description of the implementation of the nodal spans parsing system.

6.2. Restructuring the Assertion String.

The usual Immediate Constituent analysis of a sentence divides the sentence into a Subject and a Predicate, the Predicate into a Verb and an Object. String analysis treats Subject, Verb, and Object as slots to be filled all on the same level. The requirements of the nodal spans algorithm forces a restructuring of the Assertion string in the direction of Immediate Constituent analysis.

In Chapter 4 it was seen that the nodal spans algorithm must treat a grammar internally as if it were binary. Normally this can be accomplished in a straightforward manner by leftward grouping. That is, the production

A → B C D E

is treated as though it were the productions

$$A \rightarrow A_3\ E\ , \quad A_3 \rightarrow A_2\ D\ , \quad A_2 \rightarrow B\ C\ .$$

In the case of the assertion string however this grouping is not feasible.

There are three broad categories of restrictions operating on the subject, verb, and object of the assertion string:

1. Constraints between the verb and the object. These include the OBJLIST restrictions considered below, and the NOTNOBJ restrictions which verify that the object has no attributes which are prohibited for that verb. These restrictions require that the cores of the verb and the object be available at the time the two elements are concatenated.

2. Constraints between the subject and the verb, especially if the verb is something other than "to be" or a be-replacer ("seems", etc.). These include not only number agreement, but more importantly the NOTNSUBJ restrictions which check that the subject does not have illegal attributes for that verb. These restrictions require that the cores of the verb and the subject be available at the time they are joined.

3. Constraints between the subject and the object-of-be when the verb is "to be" or a be-replacer. Examples include restrictions governing the appearance of certain sentence nominalizations in the subject and in the object-of-be. It seems most convenient to check number agreement between

subject and verb by means of a flag and have the core of the subject and the object-of-be available at the time the two elements are joined.

These requirements force the verb and the object to be grouped together. The restrictions of the first category are checked first. If the verb is not "to be" or a be-replacer, the verb is taken as the core of the verb phrase and restrictions of the second category are applied when the verb phrase and the subject are concatenated. If the verb is "to be" or a be-replacer, the object-of-be is taken as the core of the verb phrase, a number flag is set from the verb, and restrictions of the third category are executed when the verb phrase and subject are concatenated. Thus the production for the assertion string becomes

<ASSERTION> → [<SA>] [<SUBJECT>] [<SA>] <ASSERTIONVO>

<ASSERTIONVO> → <VERB>[<SA>][<OBJECT>][<RV>][<SA>]

DVOB restrictions (see Appendix B) govern the choice of the core element in <ASSERTIONVO>.

This analysis of the assertion string is so prevalent[1] that it would be surprising indeed if we were to discover otherwise.

The other Subject - Verb - Object strings have also been altered in line with these considerations:

---

[1] Leonard Bloomfield, Language, New York, 1933, p. 161; Noam Chomsky, Syntactic Structures, The Hague, 1957, p. 26; H. A. Gleason, Linguistics and English Grammar, New York, 1965, p. 141.

<YESNOQ> → [<SA>]<VERB1><SUBJECT>[<RW>][<SA>][<YESNOQVO>]

<YESNOQVO> → [<VERB2>][<OBJECT>][<RV>][<SA>][<ORNOT>]

<TSUBJVO> → <NULTSUBJVO>[<SA>]<VERB1><SUBJECT>[<SA>][<TSUBJVOVO>]

<NULTSUBJVO> → NEITHER / NOR / <DSTG> / <PN>

<TSUBJVOVO> → [<VERB2>][<OBJECT>][<RV>][<SA>]

6.3. Restructuring the Left Adjunct of the Noun.

The left adjunct of the noun together with the noun variant, <LN> <NVAR>, are at present structured as suggested by Harris:[2]

((<TPOS> <QPOS> <APOS> <NSPOS> <NPOS>) <NVAR>) .

The usual binary structure given in Immediate Constituent analysis[3] is indicated by the parenthesizing:

(<TPOS>(<QPOS>(<APOS>(<NPOS> <NVAR>))))

The requirements of the nodal spans algorithm and the restrictions suggest a restructuring different from either of these.

First of all, <NSPOS> and <NPOS> are subsumed under <NPOS>. It is possible to have an <NPOS> precede an <NSPOS> as illustrated by the example

"a neighborhood fireman's ball"

This is a fairly rare construction, so it is marked as such (by a minus sign) in the revised grammar:

---

[2] Zellig Harris, String Analysis of Sentence Structure, The Hague, 1962, p. 36

[3] H. A. Gleason, op. cit., pp. 151-9.

<NPOS> → [<*NS>] [<NPOS1>] / - <NPOS1> <*NS>

<NPOS1>→ [<LCDN>] <NULNPOS> / <NPOS> [<LCDN>] <NULNPOS>

<NULNPOS> → <*N> / <*VING>

<LCDN> → *ADJ >/ - <*Q>

It is always the noun <NPOS1> which is brought up as the core.

The one class of restrictions which cannot be handled by means of flags, but must have the core lexical items at their disposal are the selectional restrictions. For example, if the core of the noun variant <NVAR> is a present participle of a verb and the <NPOS> is filled by a noun, the verb and noun must be a valid verb-object combination. If the phrase

"mud slinging"

is encountered, the parser must check the NOTNOBJ attribute of "sling" and the attributes of "mud" to see that mud is something that can be slung. To do this with flags would require us to define a flag for every possible attribute of a noun and set a flag for every attribute of the particular noun. Clearly it is more reasonable to have the core lexical item of <NPOS> available as CORE at the time an element containing <NPOS> is joined to <NVAR>. Thus this concatenation is performed first:

<NNVAR> → [<NPOS>]<NVAR>

When the adjective position is occupied by the present participle of a verb and the noun variant by a noun, for

example in the phrase

"the running boy"

a selectional restriction must check that the noun is a legal subject for the verb. This involves searching the list of noun attributes under the NOTNSUBJ attribute of the verb and the list of attributes for the noun to make sure they contain no attribute in common. Again, this cannot be done conveniently with flags, so the core of <APOS> and the core of <NVAR> both have to be available when the two elements are concatenated into the same string. Thus, the core of <NVAR> is taken as the core of <NNVAR>, and

<ANNVAR> → [<APOS>]<NNVAR>

There are number agreement and other restrictions between the article position <TPOS> and the quantifier position <QPOS>. They are not selectional restrictions so it would be possible to handle them with flags, yet it seems simpler and more compelling to group <TPOS> and <QPOS> together. Therefore, we rewrite the productions

<TQPOS> → [<TPOS>][<QPOS>]

<LNNVAR> → [<TQPOS>]<ANNVAR>

There are several restrictions which require number agreement between the article-quantifier positions and the noun. Thus, the core of <NNVAR> is taken as the core of <ANNVAR> and a number flag is set in the article-quantifier positions so that the necessary comparisons can be made.

In the phrase

"these three running boys"

the number of "these" and "three" are checked against each other, the selectional restriction between "boys" and "running" is checked, and finally the number restriction between the article and/or quantifier and the noun is checked. Thus the structure of the noun with its left adjunct is

((<TPOS><QPOS>)(<APOS>(<NPOS><NVAR>))) [4]

Null options are extremely inconvenient to handle in a nodal spans parser, so the <*NULLN> option for <NVAR> has been eliminated by the following rewriting of the production:

<LNR> → <LNNVAR>[<RN>] / <LN1> [<RN>]

<LN1> → [<TPOS>] [<QPOS>] [<APOS>]

This also eliminates the need for the restriction banning an <NPOS> when <NVAR> is null.

6.4. Adverbial Preposition - Preposition Strings.

Each span contains a reference to only one core lexical item. There are several reasons why this is desirable. It keeps the restriction metalanguage, and therefore the mechanisms implementing it, simple. Theoretically making one core word a determiner of the spans makes the n^3 nodal spans algorithm into an n^4 algorithm, and inclusion of another core word would make it n^5. More practically, it could prevent a certain amount of merging and thus cause difficulties with space. Furthermore there are very few

[4] It is interesting to note that Rulon Wells concurs with this analysis in "Immediate Constituents," Language, Vol. 23, pp. 81-117, 1947.

places in the grammar where more than one core word is called for, that is, where a restriction must look at the attributes or literal of more than one lexical item in a string.

A small class of exceptions is the class of adverbial preposition - prepositional strings, e.g. <DP1PN>: "One must face up to serious problems." A given verb which takes a <DP1PN> object may do so only when the values of the adverbial preposition and the preposition are particular words. These allowed words are stored in the dictionary entry of the verb. Thus, when a <DP1PN> object is to be concatenated with a verb to form a verb phrase, a restriction must check to see that both the adverbial preposition and the preposition are allowed, i.e. two core lexical items must be checked.

The escape from this difficulty is the fact that the class of adverbial prepositions is small -- fourteen words ("about", "across", "along", "around", "away", "back", "down", "in", "off", "on", "out", "over", "through", "up"). Thus, at very little extra cost there may be a flag corresponding to each adverbial preposition. At a point in the grammar where an adverbial preposition occurs in a DP - P string, there are fourteen restrictions of the form

IF CORE2 HAS "UP" THEN CORE HAS UPFLG.

Where a DP - P string object is to be concatenated with a verb there are fourteen restrictions of the form

IF CORE2 HAS UPFLG THEN CORE1 HAS OBJLIST:DP1PN:DPVAL:"UP".

Of course this gives the restrictions an unaesthetic appearance, and gives one reading them the feeling that it could all be compressed somehow. Nevertheless, the cost is less than if the mechanism to compress it had to be built into the metalanguage and the parser. Elegance in the restrictions is sacrificed for elegance in the metalanguage.

6.5. OBJLIST Restrictions.

As stated before, the object carries the greatest weight in the grammar. There are fifty-six alternatives which may fill the object slot. Of course, only a few of these may actually occur with any given verb. Stored with each verb in the dictionary is an attribute called OBJLIST which lists the alternatives of <OBJECT> which that verb may take. The top-down parser attempts to match only those options of <OBJECT> listed in OBJLIST. This not only executes the restriction but is a great time-saving device as well.

In the sentence

"They see flying planes."

the bottom-up parser interprets the pharse "flying planes" as a <NSTGO> and as a <VINGO>. Without OBJLIST restrictions, these two spans are merged under one <OBJECT> span. When <OBJECT> is to be concatenated with the verb a restructuring of the span collection becomes necessary to eliminate the <VINGO> interpretation.

In the bottom-up parser, there are two possible means of implementing the OBJLIST restrictions -- the principled way and the sensible way. The principled way is to have a flag corresponding to each option for object. When an <OBJECT> span is generated from the <NSTGO> node spanning "flying planes", a NSTGOFLG is set; when an <OBJECT> span is generated from the <VINGO> node, a VINGOFLG is set. The two <OBJECT> nodes are not merged since they have different flags. When the <OBJECT> spans are to be concatenated with a verb whose core word is "see", OBJLIST restrictions of the form

IF CORE2 HAS NSTGOFLG THEN CORE1 HAS OBJLIST:NSTGO.

IF CORE2 HAS VINGOFLG THEN CORE1 HAS OBJLIST:VINGO.

admit the former and reject the latter.

The first difficulty with this solution is that it increases the number of flags by about fifty percent, and thus the space that must be set aside for storing the cores. Yet, since only one of these flags may be set at one time, most of this extra space is wasted. One could have another type of flag in the metalanguage. When a restriction calls for this type of flag to be set, instead of setting a bit, a pointer is stored in the core to a list of the object alternatives. The nearly sixty bits that would have been required is now reduced to six. The difficulty with this solution is that it requires an extension of the metalanguage to cover the new flag type and corresponding additional mechanisms in the parsing program. Furthermore, there is

still the unaesthetic list in the restrictions of almost sixty nearly identical restrictions.

The sensible solution is simply to have a subroutine in the parsing program block merging under <OBJECT> nodes and to have another check in each verbal - object string to see if the object is of the proper sort for the verb. This solution is easy enough, but it is unprincipled because the very important OBJLIST restrictions would not appear in the grammar, which ought to be a true specification of English grammar as well as a module in a parsing system.

We merely note that the expression of the OBJLIST restrictions is within the scope of the metalanguage, and adopt the sensible solution. The algorithm specified in Section 6.8 includes procedures to check <OBJECT> alternatives independently of the restrictions.

None of these solutions captures the time-saving properties of the OBJLIST restrictions in the top-down environment. The mechanism in the parsing program for building up the candidates lists must be altered. A procedure is included in the algorithm of Section 6.8 which when generating a candidates list, tests for <OBJECT> and instead of adding F(<OBJECT>) to the candidates list, adds only F of those elements in OBJLIST(CORE1). It is expected that this will cut down significantly on the number of elements in the candidates lists and thus on the number of spans generated.

There is also stored in the dictionary entry for each verb the attribute POBJLIST which lists the strings which may appear as the passive objects of the verb when it appears in the passive form. The POBJLIST restrictions are handled in a way similar to the OBJLIST.

6.6. Writing Restrictions.

In this section it will be shown how several restrictions of nondecreasing complexity can be coded in the restriction metalanguage. This sho ld at the same time impart the flavor of the restrictions and demonstrate the use of the metalanguage.

WPOS1A: Adverbs can appear in many adjunct positions, but each adverb may appear in only certain positions. Thus "generally" may occur asa sentence adjunct while "very" may not. There is only one adverb category, so the WPOS1 restrictions check the attributes of the adverb to see if it may legally occupy that position. An adverb has attributes which correspond to the positions it may occur in; thus an adverb which may appear as a sentence adjunct has the attribute "DSA" and an adverb that may appear as the right adjunct of a noun has the attribute "DRN". Restriction WPOS1A checks that an adverb has attribute "DSA" before accpeting it in sentence adjunct position.

HOUSED AT <DSTG>: <REPSA>: [5]

$POS4 [6] OR CORE2 HAS DSA.

---

[5] <REPSA> is a nonterminal symbol allowing for repeated sentence adjuncts.

[6] The semigrammaticality classes are explained in Section 7.3.

As defined in the BNF component, the core of a string of adverbs <DSTG> is the last adverb in the string.

WN1: The article in the article position in the left adjunct of the noun may be preceded by a quantifier, but only certain quantifiers may occur with certain articles. Thus, we may say "such a" but not "such the" and "all the" but not "all a". The relevant productions are

<LTR> → [<LT>] <*T> / <*D> .

<LT> → <*D> / <*Q>

When the quantifier is matched as the left adjunct of the article, a check is made that the quantifier is of the right sort:

```
HOUSED AT <LT>: <LTR>:
$POS4 OR IF CORE2 HAS Q THEN CORE2 HAS QHALF OR QALL OR QMANY.
```

When the article position is matched, a check is made that the quantifier and article are compatible:

```
HOUSED AT <* T>: <LTR>:
$POS4 OR IF CORE1 HAS QMANY THEN CORE2 HAS "A" OR "AN".
$POS4 OR IF CORE1 HAS QALL THEN CORE2 HASNT "A" OR "AN".
$POS4 OR IF CORE1 HAS Q THEN CORE2 HASNT "NO" OR TQUAN.
```

If there is no left adjunct the restrictions succeed vacuously. The quantifiers "many" and "such" have the attribute QMANY. "All" and "both" have QALL. "Half", "double", "triple", etc. have the attribute QHALF. The articles with the attribute TQUAN are "some", "either",

"neither", "any", and "each".

DN52: the only pronouns which can take an appositive are "you" and "we", i.e. those which are plural and definitely human. Thus we may say "we the people" but not "he the benefactor". An exception to this is in a pronoun as the object of the verb "be"; we may say "It was he, the benefactor of all mankind". The relevant productions are

<REPRN> → ... / <APPOS> / ...

<RN> → <REPRN> [<RN>]

<LPROR> → [<LPRO>] <*PRO> [<RN>]

<NSTG> → ... / <LPROR>

<OBJBE> → ... / <NSTG> / ...

In top-down parsing, this constraint can be executed by a restriction called as <REPRN> is about to take the alternative <APPOS>. A locating relation travels up the tree, which is already built, to see if <RN> is occurring in <LPROR>. If so, another locating relation travels farther up the tree to see if <LPROR> is occurring in <OBJBE>. In bottom-up parallel parsing, the restriction cannot be executed at the occurrence of <APPOS> in <REPRN> for the tree above the <APPOS> node has not yet been constructed. It cannot be executed entirely at the <LPROR> or the <OBJBE> nodes since by then several other spans could have been merged under <REPRN>. First we must set a flag to prevent other spans from merging under the span for <REPRN> along with <APPOS>.

HOUSED AT <APPOS>: <REPRN>:

CORE HAS APPOSFLG.

This information must then be passed up to the level at which it will be used.

HOUSED AT <REPRN>: <RN>:

CORE HAS APPOSFLG OF CORE2.

The first half of the restriction is then applied.

HOUSED AT <RN>: <LPROR>:

IF CORE2 HAS APPOSFLG AND CORE1 HASNT NHUMAN AND PLURAL

THEN CORE HAS ONLYOBFLG.

ONLYOBFLG must be explicitly passed up to <NSTG> from <LPROR>.

HOUSED AT <LPROR>: <NSTG>:

CORE HAS ONLYOBFLG OF CORE2.

If the ONLYOBFLG is set, then <LPROR> can occur only as an object of "be". All other paths are blocked.

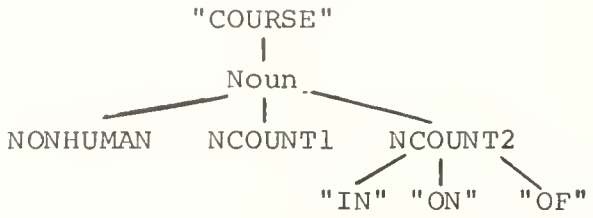
HOUSED AT <LPROR>: ALL OCCURRENCES EXCEPT <NSTG>;

<NSTG> : ALL OCCURRENCES EXCEPT <OBJBE>:

$POS4 OR CORE2 HASNT ONLYOBFLG.

WN9: A count noun, attribute NCOUNT1, requires a preceding article ("a broom", "the broom"), wh-word ("which broom"), quantifier ("one broom"), or possessive noun ("Samantha's broom"). However certain count nouns, those with attribute NCOUNT2, can occur in a prepositional phrase with specified prepositions. Thus "course" is a

count noun, but it may occur without a left adjunct following "of", "in", and "on". The structure of the lexical item for "course" is



A second exception to the count noun rule is that certain count nouns those with attribute NCOUNT3, can occur as an object of be without a left adjunct. Thus we may say "You are boss" even though "boss" is a count noun. The relevant productions are

<TPOS> → <LTR> / <WHLN> / <LNSR> / <HOWQSTG>

<TQPOS> → [<TPOS>] [<QPOS>]

<LNNVAR> → [<TQPOS>]<ANNVAR>

<LNR> → <LNNVAR>[<RN>] / ...

<NSTG> → <LNR> / ...

<NSTGO> → <NSTG>

<PN> → [<LP>]<*P>[<NSTGO>]

<OBJBE> → <NSTG> / ...

It is first necessary to set a flag if a valid left adjunct occurs that will not be carried up as the core of <TQPOS>.

HOUSED AT <WHLN>: <TPOS>:

CORE HAS OKCOUNTFLG.

HOUSED AT <QPOS>: <TQPOS>:

IF CORE2 HAS Q AND SINGULAR AND QNUMBER THEN CORE HAS OKCOUNTFLG.

When the noun is concatenated with the article, if the noun is singular, NCOUNT1 but not NCOUT2 or NCOUNT3 then the article position must be checked for the proper form

```
HOUSED AT <ANNVAR>: <LNNVAR>:
$WS OR IF CORE2 HAS SINGULAR AND NCOUNT1 AND NOT
    NCOUNT2 AND NOT NCOUNT3 THEN CORE1 HAS T OR NS
    OR OKCOUNTFLG.
```

At the same point, if the noun is NCOUNT2 and fails to have the proper left adjunct, a flag must be set to indicate that it can only occur in a prepositional phrase.

```
IF CORE2 HAS SINGULAR AND NCOUNT1 AND NCOUNT2 AND
    CORE1 HASNT T OR NS OR OKCOUNTFLG THEN CORE
    HAS ONLYPNFLG.
```

Similarly, if the noun is NCOUNT3 and fails to have the proper left adjunct, a flag is set to indicate that it can only occur as an object of "be".

```
IF CORE2 HAS SINGULAR AND NCOUNT1 AND NCOUNT3 AND CORE1
  HASNT T OR NS OR OKCOUNTFLG THEN CORE HAS ONLYOBFLG.
```

ONLYPNFLG and ONLYOBFLG must be passed up as many levels as they will be used, and must be used to block all constructions except the specified ones.

```
HOUSED AT <LNNVAR>: <LNR>;
          <LNR>:    <NSTG>:
CORE HAS ONLYOBFLG, ONLYPNFLG OF CORE2.
```

HOUSED AT <LNR>: ALL OCCURRENCES EXCEPT <NSTG>;

<NSTG>: ALL OCCURRENCES EXCEPT <OBJBE>; <NSTGO>:

$WS OR CORE2 HASNT ONLYOBFLG OR ONLYPNFLG.

We must take care of the case where a noun is both NCOUNT2 and NCOUNT3, so that it may occur as an object of "be" or in a prepositional phrase.

HOUSED AT <NSTG>: <OBJBE>:

$WS OR IF CORE2 HAS ONLYPNFLG THEN CORE2 HAS ONLYOBFLG.

HOUSED AT <NSTG>: <NSTGO>:

$WS OR IF CORE2 HAS ONLYOBFLG THEN CORE2 HAS ONLYPNFLG.

ONLYPNFLG must be passed the rest of the way up to the prepositional phrase and used to block all other constructions.

HOUSED AT <NSTG>: <NSTGO>:

CORE HAS ONLYPNFLG OF CORE2.

HOUSED AT <NSTGO>: ALL OCCURRENCES EXCEPT <PN>:

$WS OR CORE2 HASNT ONLYPNFLG.

Furthermore, in the prepositional phrase a check must be made for the proper preposition for the count noun.

HOUSED AT <NSTGO> : <PN>:

$POS5 OR IF CORE2 HAS ONLYPNFLG THEN CORE1 EL NCOUNT2(CORE2).

6.7. Examples of Restrictions in Action.

In this section all the restrictions which are housed at one particular point in the grammar are listed, their intent is briefly explained and their operation described. Like the

last section, this should impart the flavor of the restrictions and shed light on their operation in the course of parsing. The particular point in the grammar chosen because of its importance is the occurrence of the verb phrase <ASSERTIONVO> in <ASSERTION>. Probably more restrictions act at this point and more flags are active than at any other point in the grammar. The relevant productions are

<ASSERTION> → [<SA>][<SUBJECT>][<SA>]<ASSERTIONVO>

<ASSERTIONVO> → <VERB>[<SA>][<OBJECT>][<RV>][<SA>]

We will be looking at all restrictions housed at <ASSERTIONVO>: <ASSERTION>. If at any point a restriction fails, an exit is made from the testing routine and the new span is not formed (unless in semigrammaticality mode, as explained in Chapter 7).

A. Deepest Verb and Object of "Be" Restrictions.

DVOB5: The deepest verb and deepest object of "be" restrictions determine whether the deepest verb or the deepest object of "be" is taken as the core, as explained in Section 6.2. If the deepest verb, the DVBFLG has been set; if the deepest object of "be", the DOBFLG.
Since the core of the verb phrase becomes the core of the assertion these flags must also be passed up to tell which element the core is. In addition, for the sake of condensation in the grammar, the number flags SINGFLG and PLURALFLG and a flag indicating that the object of "be" is a string of adjectives ASTGOBFLG are passed up in the same restriction.

CORE HAS DVBFLG, DOBFLG, ASTGOBFLG, SINGFLG, PLURALFLG OF CORE2.

If these flags are set in the core of <ASSERTIONVO> then they are set in the core of <ASSERTION>.

B. Agreement Restrictions.

WAGREE1: By the time <ASSERTIONVO> is concatenated with <SUBJECT> both spans have their number, if they have a number, either as an attribute in the core lexical item or as one of the flags SINGFLG or PLURALFLG. Thus, the restriction checking compatible numbers is

$SS OR IF CORE2 HAS SINGULAR OR SINGFLG THEN CORE1 HASNT PLURAL OR PLURALFLG.

$SS OR IF CORE2 HAS PLURAL OR PLURALFLG THEN CORE1 HAS NOT (SINGULAR OR SINGFLG) OR AGGREGATE OR "I".

WAGREE2: Except in rare cases, the subject and object of "be" agree in number:

$RARE OR IF CORE1 HASNT "IT" AND CORE2 HAS DOBFLG THEN (IF CORE2 HAS (SINGULAR OR SINGFLG) AND NOT AGGREGATE THEN CORE1 HASNT(PLURAL OR PLURALFLG) AND(IF CORE2 HAS PLURAL OR PLURALFLG THEN CORE1 HAS NOT (SINGULAR OR SINGFLG) OR AGGREGATE).

WAGREE3: If a collective verb has no object then the subject cannot be singular. Thus, we may say "The chemicals combine." but not "The chemical combines." If a reciprocal verb has no object, then the subject cannot be singular. Thus, we say "He and I correspond." but not "I correspond."

If the verb phrase has no object, NULLOBJFLG will be set in the core of <ASSERTIONVO>.

$WS OR IF CORE2 HAS NULLOBJFLG AND VCOLLECTIVE THEN CORE1 HAS NOT (SINGULAR OR SINGFLG) OR AGGREGATE OR COLLECTIVE.

$WS OR IF CORE2 HAS NULLOBJFLG AND VRECIP THEN CORE1 HAS NOT (SINGULAR OR SINGFLG) OR AGGREGATE.

WAGREE6: An assertion which appears in the sentence adjunct wh-string <SAWHICHSTG> ("which surprised us") cannot have a plural verb. Thus the PLURALFLG must be passed up from <ASSERTIONVO> to <ASSERTION>.

CORE HAS PLURALFLG OF CORE2.

WAGREE7: The subject must agree in number with its right adjunct appearing to the right of the object. Thus we may say "John threw the ball himself." but not "John threw the ball themselves." If a singular or plural right adjunct of the subject occurs to the right of the object then RSSINGFLG or RSPLURFLG are set in the core of <ASSERTIONVO>, respectively.

$SS OR IF CORE2 HAS RSSINGFLG THEN CORE1 HASNT PLURAL OR PLURALFLG.

$SS OR IF CORE2 HAS RSPLURFLG THEN CORE1 HAS NOT (SINGULAR OR SINGFLG) OR AGGREGATE.

C. Comma Restrictions.

DCOM2: There cannot be a comma between a verb and its immediately preceding subject. Thus if there is a comma before the verb, there must be an intervening adjunct. If there is a sentence adjunct in the position <ASSERTION>, 3 or if the subject has a right adjunct, INTADJFLG has been set in $<\text{ASSERTION}>_2$ or $<\text{ASSERTION}>_3$.

$PUNC OR IF INTER(P) = "," THEN CORE1 HAS INTADJFLG.

D. Selectional Restrictions.

WSEL2: If the subject is a noun or pronoun and the verb is active then there must be selectional agreement between them. This is determined by the NOTNSUBJ list-attribute of the verb which lists the attributes its subject may not have.

$SEL1 OR IF CORE1 HAS N OR PRO AND CORE2 HAS DVBFLG
AND NOTNSUBJ AND NOT DVBVENPASSFLG THEN
NOTNSUBJ(CORE2) * CORE1 = NL.

Furthermore, if the subject is the expletive "there" the verb phrase has an object of "be".

$SEL1 OR IF CORE1 HAS "THERE" THEN CORE2 HAS DOBFLG.

E. Sentence Nominalization Restrictions.

There are eleven sentence nominalization restrictions which act at this point in the grammar.

DSN1: If the object of "be" is a sentence nominalization

or an assertion then the subject must be a sentential noun, NSENT1, NSENT2, or NSENT3, ("The fact is that...", "The evidence is that..." "The trouble is that...") or a nonhuman pronoun ("it"), or a noun string with a zeroed noun or a <FORTOVO> or <TOVO> string. If the object of "be" is a sentence nominalization or an assertion, SNAOBFLG is set in the core of <ASSERTIONVO>. If the subject is a noun phrase with a zeroed noun, NULLNFLG is set in the subject. If the value of the subject is <FORTOVO> or <TOVO> then the FORTOVOFLG or TOVOFLG is set, respectively.

$POS4 OR IF CORE2 HAS SNAOBFLG AND SPAN1 ISNT NL THEN CORE1 HAS NSENT1 OR NSENT2 OR NSENT3 OR (PRO AND NOT NHUMAN) OR NULLNFLG OR FORTOVOFLG OR TOVOFLG.

DSN2: If a sentence nominalization or an assertion appears as the right adjunct of an adjective in the object, then the subject cannot be a sentential noun, a sentence nominalization, a present participial string or a noun string beginning with a wh-word. Thus we cannot say "The fact is true that he came." If a sentence nominalization or an assertion appears as the right adjunct of the object position adjective, the core of <ASSERTIONVO> will have SNARAFLG. If the value of the subject is a sentence nominalization, a present participial string, or a noun string beginning with a wh-word, the core of the subject will have SUBJSNFLG, SUBJVINGFLG, or SUBJWHFLG, respectively.

$POS4 OR IF CORE2 HAS SNARAFLG THEN CORE1 HASNT SUBJSNFLG OR SUBJVINGFLG OR SUBJWHFLG OR NSENT1 OR NSENT2.

DSN4: If a sentence nominalization occurs as the right adjunct of the subject in the post-object position, RNSNFLG is set and passed to the core of <ASSERTIONVO>. If this is the case, the subject must be a sentential noun and cannot have a sentence nominalization as its immediate right adjunct. For example, we may say "Evidence was presented that the president was guilty." or "Evidence that the president was guilty was presented." If the immediate right adjunct of the subject is a sentence nominalization, the RNSNFLG is set in the core of the subject.

$POS4 OR IF CORE2 HAS RNSNFLG AND SPAN1 ISNT NL THEN CORE1 HAS (NSENT1 OR NSENT2) AND NOT RNSNFLG.

DSN5: Several situations in the verb phrase require the subject to be "it" and in these cases SUBJITFLG is set. Then when the verb phrase is to be concatenated with the subject, the following restriction is applied.

$POS4 OR IF CORE2 HAS SUBJITFLG THEN CORE1 HAS "IT".

DSN7: If <C1SHOULD> occurs as the right adjunct of the subject in the post-object position, then the C1SHOULDFLG is set and passed to the core of <ASSERTIONVO>. In this case, the subject must be in the subclass of sentential nouns with the ASHOULD attribute. For example, we may say "The prayer is on the lips of all that he be safe and sound."

$POS4 OR IF CORE2 HAS ClSHOULDFLG AND SPAN1 ISNT NL
THEN CORE1 HAS NSENT1:ASHOULD.

DSN9: <TOVO> may appear as the right adjunct of an adjective sententially ("John was diligent to have solved the problem.", "It was easy to solve the problem.") and nonsententially ("John was able to solve the problem.") When it appears nonsententially in the object position, RATOVOFLG is set and passed to the core of <ASSERTIONVO> and in this case the subject must be human.

$POS4 OR IF CORE2 HAS RATOVOFLG AND SPAN1 ISNT NL THEN
CORE1 HAS ((N AND (NHUMAN OR AGGREGATE)) OR
(PRO AND (NHUMAN OR PLURAL)).

DSN11: The verb in the verb phrase may be a tensed verb <LTVR>, a tense word followed by a tenseless verb <LWR><LVR>, or simply a tenseless verb <LVR> (in subjunctive constructions). In these cases TVFLG, WVFLG, or VFLG is set, respectively. <ASSERTION> occurs in <ClSHOULD> and in <SVO> and certain of these verb options are compatible with each case. In any event, the flags must be passed from <ASSERTIONVO> up to <ASSERTION>.

CORE HAS TVFLG, WVFLG, VFLG OR CORE2.

WSN1: If the subject is a sentence nominalization, the verb must be "be" or a be-replacer (indicated by a DOBFLG in the core of <ASSERTIONVO>) or a sentential verb VSENT1 or VSENT2 ("That he came surprised us", "That he came proves he cares") or must occur in certain special expressions VEXP

("That he came drove me out of my mind.") Certain sentential verbs VSENT3 can have a sentence nominalization as the subject if the verb is in the passive ("That he came is known to me.") If the deepest verb is passive the core of <ASSERTIONVO> has DVBVENPASSFLG.

$POS4 OR IF CORE1 HAS SUBJSNFLG THEN CORE2 HAS DOBFLG OR VSENT1 OR VSENT2 OR VSENT3 OR VMOD OR VEXP.

$POS4 OR IF CORE1 HAS SUBJSNFLG AND CORE2 HAS VSENT3 THEN CORE2 HAS DVBVENPASSFLG.

WSN2: If the subject is a sentence nominalization and the deepest object of "be" is an adjective or a present participle, then the adjective or present participle must be sentential, i.e. must have the attributes ASENT1 or ASENT2 or VSENT1. Thus we may say "That he came here is probable." and "That he came here is alarming."

$POS4 OR IF CORE1 HAS SUBJSNFLG AND CORE2 HAS DOBFLG AND ASTGOBFLG AND (ADJ OR VING) THEN CORE2 HAS ASENT1 OR ASENT2 OR VSENT1.

WSN3: If the object of "be" is the <FORTOVO> string then the subject must be also. If the object of "be" is the <TOVO> string then the subject must be also. TOVOFLG and FORTOVOFLG will have been set in the cores of the subject and the verb phrase when the concatenation is about to take place.

$POS3 OR IF CORE1 HAS FORTOVOFLG AND CORE2 HAS DOBFLG
AND SNAOBFLG THEN CORE2 HAS FORTOVOFLG.

$POS3 OR IF CORE1 HAS TOVOFLG AND CORE2 HAS DOBFLG
AND SNAOBFLG THEN CORE2 HAS TOVOFLG.

WSN4: If the verb has "should" as its tense word, this must be known by the parser when the span for <ClSHOULD> is formed from the span for <ASSERTION>. Therefore SHOULDFLG is set at <LWR> in <VERB> and must be passed from the core of <ASSERTIONVO> to the core of <ASSERTION>.

CORE HAS SHOULDFLG OF CORE2.

F. Verb and Center Restrictions.

DVC1: When a span for <VERB> is matched with <ASSERTIONVO>, the NULLOBJFLG is set. When a span for <OBJECT> is matched with <ASSERTIONVO>, this flag is unset. Since <OBJECT> is optional, <ASSERTIONVO> may have only the verb. However in this case, the verb must be intransitive, i.e. NULLOBJ must be one of the possible options on its OBJLIST. Therefore at all occurrences of <ASSERTIONVO> (in fact, there is only one), if there is no object, OBJLIST of the verb must be checked.

$POS2 OR IF CORE2 HAS NULLOBJFLG THEN NULLOBJ EL
OBJLIST(CORE2).

This is the only OBJLIST restriction which is included in the bottom-up restriction component.

WVC8: If the object of "be" is a prepositional phrase and the preposition is "since" then the subject must be "it". Thus we may say "It is since Sunday that they have been visiting." but not "Their visit is since Sunday."

$POS4 OR IF CORE2 HAS DOBFLG AND "SINCE" THEN CORE1 HAS "IT".

G. Wh-String Restrictions.

WWH2: Null alternatives in productions are very difficult to handle in nodal spans parsing. Therefore they have been eliminated from the grammar in favor of optional elements in strings. The subject of an assertion can be omitted in certain wh-strings, as in "The man who () came to dinner." Here the assertion is "() came to dinner." Thus, <SUBJECT> is marked as optional in the production for <ASSERTION>. But assertions with omitted subjects can occur only in certain constructions. Hence the omission of the subject must be detected and a flag set to signal this fact. This is accomplished by the following restriction housed at both <SA> of <ASSERTION>, 3, and <ASSERTIONVO> at <ASSERTION>,4:

IF SPAN1 IS NL THEN CORE HAS NULLSUBJFLG.

WWH3A: Similarly if an object position is not filled it may be because the object is omitted in a wh-string, as in the sentence "This is the book which I read ()." When this is detected in the <ASSERTIONVO> string, the NULLWHNOFLG is set. The object omission may be embedded deeply in a

sequence of assertion-like strings, as in the sentence "This is the book which I think I told you that I read ()." The information that an object has been omitted is transmitted to higher levels by means of the OOMITFLG. Thus the NULLWHNOFLG is traded in for an OOMITFLG at the earliest possible opportunity and one of these opportunities is at <ASSERTIONVO> in <ASSERTION>:

IF CORE2 HAS NULLWHNOFLG THEN CORE HAS OOMITFLG.

WWH4: When an omitted object of "be" is detected in <ASSERTIONVO> a NULLOBJBEFLG is set, and is traded in at the earliest possible opportunity for a SOMITFLG which carries the information to higher levels. Thus we have the restriction

IF CORE2 HAS NULLOBJBEFLG THEN CORE HAS SOMITFLG.

WWH5: When an omitted prepositional phrase is detected in the object position, as in the sentence "He is a man on whom we can rely ()." a NULLWHPOFLG is set, and is traded in at the earliest possible opportunity for a POMITFLG which carries the information to higher levels. Thus we have the restriction

IF CORE2 HAS NULLWHPOFLG THEN CORE HAS POMITFLG.

WWH1A: If SOMITFLG, OOMITFLG and POMITFLG occur in the core of <ASSERTIONVO>, as for example OOMITFLG would occur in the core of the verb phrase "think I thold you that I read ()", they must be passed on to higher levels. This

is done with the restriction

CORE HAS SOMITFLG, OOMITFLG, POMITFLG OF CORE2.

WWH1C: No string may have more than one omitted element. The assertion string may cause difficulties in this regard for the subject and the object could both contain omitted elements. Thus there must be a restriction housed at <ASSERTIONVO> in <ASSERTION> which blocks this possibility. This restriction states that if CORE1 contains any flag which indicates an omission then CORE2 cannot, and conversely. The restriction will not be written out here as it is bulky; the reader is referred to Appendix B.

WWH53: For <NUL2WHQ-N> to take the <ASSERTION> option instead of the <YESNOQ> option, the verb cannot be "be", "have", or "do". Thus when <VERB> is matched with <ASSERTIONVO>, if its core has an attribute VBE, VHAVE, or VDO, the VHAVEBEDOFLG is set. This information must be passed up to <ASSERTION>. Hence the restriction

CORE HAS VHAVEBEDOFLG OF CORE2.

6.8. The Complete Algorithm

The complete parsing algorithm is essentially that of Section 4.4, but several modifications discussed above have been made. An atomic node is generated for each category or meaning of a word. Before a new span is formed a call to restrictiontest (defined in Section 5.10) is made to execute

the restrictions housed at that point in the grammar. restrictiontest returns the value pas which reports whether or not the restrictions have succeeded. restrictiontest also returns the core of the new span is newcore if the restrictions succeed. The core of a node is included in the span as one of its determiners and is extracted with the function core. In forming the new candidates lists the optimization suggested in Section 6.5 is implemented: When a verb is encountered, first of all its possible successors except 'object' is added to the list together with first of every option of object given in OBJLIST of the verb. Otherwise the algorithm and externals are those of Section 4.4.

A word is stored as a tree structure as described in Section 2.5. This tree is viewed as a set word together with three functions: literal which picks out the root node of the word, desc which returns the set of descendents for each node in the tree, and name which returns the name of a node, i.e. the literal spelling of the root node, the name of the attribute or category for the other nodes. Thus to extract the set of category nodes of a word corresponding to its different meanings, we apply the function categories defined as:

```
definef categories(word);
return desc(literal(word)); end categories;
```

In the main algorithm given below, it will be assumed that restrictiontest returns as the value pas, the integer 0

if all restrictions pass and some nonzero integer otherwise. Thus, for the moment, in the condition in line (2) we may assume #sghier is 0. This condition and all the references to the function sgval are explained in Section 7.2. For the moment, they may be ignored.

```
define parse;
/* the following are assumed to be global: sentence, sn,
     ref, root, eta, kay, jay, jayl, posinitflg, necinitflg,
     posterflg, possuc, pee, que, divlis, candswitch, first,
     core, catnode, sghier */
if candswitch then candlist(0) = first(root);
(1 ≤ ∀i ≤ sn) word = sentence(i);
   /* stack up all meanings of ith word */
   (∀cat∈categories(word)) if n candswitch or name(cat)∈
      candlist(i-1) then stack = stack with
      (<i-1,<name(cat),0,0>,i,<nl,catnode,name(literal(word))>>
      is x); divlis(x) = cat;
      sgval(x) = 0;                                          (1)
   addspan(x);; end ∀cat;
   /* for every span on stack and every occurrence of
      symbol in grammar */
   (∀span∈stack doing stack = stack less span;)
   (∀occ ∈ ref(eta(span)))
   /* for every possibly initial occurrence, generate
      a new span */
   if posinitflg(occ)
```

```
   then if n candswitch or eta(occ)∈candlist(pee(span))
   /* provided restrictions allow the new span */
      then restrictiontest(pas,newcore,nl,span,occ);
         if pas le #sghier                                    (2)
         then newsp=<pee(span),if possuc(occ) eq nl then
            <eta(occ),0,0> else <eta(occ),kay(occ),jay(occ)>,
            que(span),newcore>; divlis(newsp)={<span>};
         sgval(newsp) = max(pas,sgval(span));                 (3)
         addspan(newsp); termhandl;;;
      if necinitflg(occ) then continue ∀occ;;;
   /* for every occurrence of the symbol in the middle of
      a production, search for a matching span */
   (∀sp∈spans|(eta(occ) is et) eq eta(sp) and (kay(occ) is k)
      eq kay(sp) and (jay(occ) is j)-jayl(occ) le jay(sp)
      and jay(sp) le j-1 and pee(span) eq que(sp))
   /* execute objlist and pobjlist restrictions */
      objlisttest(pas,sp,span,occ); if pas gt #sghier then
         continue ∀sp;;
   /* form the new span if the restrictions allow */
   restrictiontest(pas,newcore,sp,span,occ);
   if pas le #sghier                                          (4)
   then newsp = <pee(sp), if possuc(occ) eq nl then
      <et,0,0> else <et,k,j>, que(span), newcore>;
     divlis(newsp) = {<sp,span>};
     sgval(newsp) = max(pas,sgval(sp),sgval(span));           (5)
     addspan(newsp); termhandl;;
end ∀sp; end ∀occ;; end ∀span; end ∀i; end parse;
```

```
macro termhandl;
   if (kay(newsp) eq 0 and jay(newsp) eq 0) then
      stack = stack with newsp;;
   /* if the new span is possibly terminal but not necessarily
      terminal, generate a necessarily terminal span and
      stack it */
   if posterflg(occ) and possuc(occ) ne nl
   then nusp = <pee(newsp),<eta(occ),0,0>,que(newsp),
      core(newsp)>;
      divlis(nusp) = {<newsp>};
      sgval(nusp) = sgval(newsp);                          (6)
      addspan(nusp); stack = stack with nusp;;
   /* form new candidates list  utilizing objlist optimization*/
   if candswitch then candlist(i) = candlist(i) +
      if eta(newsp) ∈ vostglist and eta(span) ∈ verballist
      then {'object'}+ first[possuc(occ)-{'object'}+objlist
         (catnode(core(span)))]
      else if eta(newsp)∈vpostglist and eta(span)∈verballist
      then {'passobj'}+first[possuc(occ)-{'passobj'}+pobjlist
         (catnode(core(span)))]
      else first[possuc(occ)];; endm;

macro addspan(spn);
   /* see if span has been generated already */
   if (∃spa ∈ spans|((∀f∈{pee,que,eta,kay,jay,core})
      f(spa) eq f(spn)) and eta(spn) ne 'object' and eta(spn)
      ne 'passobj')
   /* if so, modify the divisions list of the existing span */
```

```
   then divlis(spa) = divlis(spa) + divlis(spn);
      sgval(spa) = min(sgval(spa),sgval(spn));          (7)
   /* if not, add the new span to spans */
   else spans = spans with spn;  endm;
```

The routine for checking OBJLIST and POBJLIST restrictions is as follows:

```
define objlisttest(pas,span1,span2,occ); pas = 0;
if ((span1 ne nl) and (eta(span2) eq 'object')
   and (eta(occ) ∈ vostglist))
then if eta(hd(∋divlis(span2)))∈objlist(catnode(core(span1)))
   then pas=0; else pas=#sghier+1;;;
if ((span1 ne nl) and (eta(span2) eq 'passobj') and (eta(occ)∈
   vpostglist))
then if eta(hd(∋divlis(span2))) ∈ pobjlist(catnode(core(span1)))
   then pas=0; else pas=#sghier+1;;; return;
end objlisttest;
```

The sets vostglist, vpostglist, verballist, and the functions objlist and pobjlist are defined as follows:

```
vostglist = {'vo', 'vingo', 'veno', 'tovo', 'vingofn',
          'assertionvo', 'yesnoqvo', 'tsubjvovo'}
vpostglist = {'venpass'}
verballist = {'verb','lv','lvingr','lvenr','verb2','ving'}
```

```
definef objlist(cat);
return {x∈desc(y)|y∈desc(cat) and name(y) eq 'objlist'};
end objlist;
definef pobjlist(cat);
return {x∈desc(y)|y∈desc(cat) and name(y) eq 'pobjlist'};
end pobjlist;
```

6.9. The Implementation.

A nodal spans parsing system based on the Linguistic String Project's grammar is being implemented on the CDC 6600 at New York University. The system has three phases: compiling, loading, and parsing.

The compiling is done by Ralph Grishman's versatile compiler[7] which employs a top-down parser and is used now for compiling the various parts of the String Project's grammar and also for top-down parsing of English sentences. The syntax and semantics of the restriction metalanguage is specified by means of a BNF grammar extended by a small number of look-ahead and generator routines. The restrictions expressed in the metalanguage are read in, parsed and compiled by the Grishman compiler according to this metalanguage specification. The process is illustrated in Figure 1.

---

[7] Ralph Grishman, "The Implementation of the String Parser," in Courant Computer Science Symposium 8: Natural Language Processing, R. Rustin, ed., New York 1973.

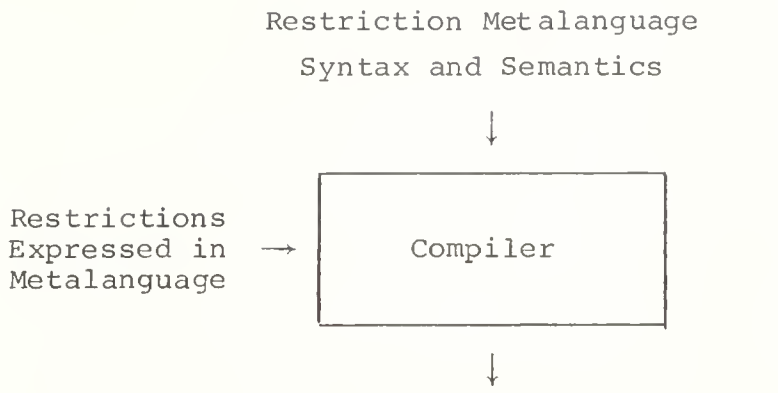


Figure 1

The loading phase begins by reading in the BNF component of the grammar and converting it to the inverse representation described in Section 4.2. The F-relationship, i.e. the set of symbols which can apper leftmost in a tree depending from a given symbol, is generated in bit matrix form for use in forming the candidates lists. The restrictions are loaded as stacks of operators in Reverse Polish notation, generated during the compilation. For example, the restriction

CORE2 HAS AFLG OR BFLG AND CORE1 HAS CFLG.

is converted by the compiler to the form

CHECK(2, AFLG)

CHECK(2, BFLG)

OR

CHECK(1, CFLG)

AND

Linked lists produced by the compiler indicate the housing specifications. These specifications are placed in the corresponding entries of the inverse grammar.

These first two phases need be repeated only after the grammar has been changed.

At the start of the parsing phase, the lexical scan and dictionary lookup routines of Grishman's system provide the parser with the dictionary entry for each word in the English sentence to be parsed. The parser itself is based on the merging parsing program used to generate the statistical results of Chapter 4. It is extended as indicated in the previous section. The routines of this parser are organized on three levels:

Level 1: routines which govern the flow of the algorithm.

Level 2: routines which deal with the data structures.

Level 3: routines involved with how information is packed into the machine words.

Thus, if the parser were to be converted to another machine, only Level 3 routines would have to be rewritten. If it were desirable to revise the data structures, only Level 2 and some Level 3 routines would need alteration. During the nodal spans parsing, calls are made to a routine which executes the restrictions. This segment of the program is based on the SETL specifications of Section 5.10. The output of the parser is a concise listing of all interpretations of the sentence, similar to the one exhibited in Section 1.4.

The semigrammaticality options described in the next chapter are also being implemented.

CHAPTER 7. AN ALGORITHMIC APPROACH TO SEMIGRAMMATICALITY.

7.1. Introduction.

One goal of the study of syntax is to identify the grammatical sentences. But the class of grammatical sentences has a fuzzy boundary, not one that can be sharply delineated. The least a theory of syntax should hope for is to identify those sentences which are unquestionably grammatical and those which are unquestionably ungrammatical. Beyond this it should impose a more or less refined gradation upon the continuous spectrum between the grammatical and the ungrammatical. Correspondingly, a computer parsing program ought to report not only whether a sentence is grammatical but also for those sentences which are "nearly" grammatical just how far from the grammatical they are. The analyses of these semigrammatical sentences should pinpoint where and for what reason the sentence fails to be grammatical.

We certainly sense that some partially grammatical sentences are closer to the grammatical than others. For example, the string

"Three men flies to London."

seems more nearly grammatical than the string

"Thinks flies to London."

and there are some who would say that

"Falsehood flies to London."

is unquestionably grammatical. One way to view semigrammati-

cality is to say that a semigrammatical sentence is a string of words which fails to be grammatical because of infractions of rules of the grammar, which however are not serious enough to destroy all resemblance to a grammatical sentence. Then the existence of degrees of semigrammaticality would imply that some rules have more force than others. For example, the rule prohibiting a tensed verb from the subject position has more force than the rule requiring number agreement between subject and verb, and both have more force than selectional constraints between the subject and verb. Therefore we ought to be able to characterize the degrees of semigrammaticality by assigning the rules to different classes in a hierarchy corresponding to their force.

The difficulty with this approach is that a good machine grammar is not necessarily a cognitive model, and the degrees of grammaticality imposed by the grammar may not correspond with our intuitive feelings about the sentences. For example, the sentence

"Scientists truth the universe."

is ungrammatical because a noun is in the verb position. The sentence

"Scientists balcony the universe."

breaks the same rule. Yet the first has power as a metaphor and hence is more acceptable than the second, which does not. The greater acceptability of the first probably cannot be captured in a natural way within anything less than a fairly

sophisticated semantic model. In general, the classification of partially grammatical sentences will suffer from the limitations of the grammar.

With this warning in mind, we may proceed with the original proposal of deriving a semigrammatical hierarchy for the given grammar.

There are three types of semigrammaticality:

1. Fragments: One or more strings of more than one word are detected in the input being analyzed, but the parser is unable to piece them together into a sentence. Examples are

"The guest who wouldn't leave in spite of everything we tried."

"In the highlands my broken heart."

2. Error: The infraction is not one the speaker would intend and may strike the listener as illiterate. Normally violations of number, count noun, and punctuation rules fall into this category, although this is not necessarily the case. One can imagine a number restriction violation with metaphorical intent, e.g.

"Wittgenstein were the two greatest philosophers of the Twentieth Century."

3. Metaphor: The infraction is intended by the speaker who is thereby trying to extend the power of the language. Most violations of selectional restrictions occurring in real, uncontrived situations are probably metaphorical in nature. If such a violation does occur out of error, the error very

likely goes deep, into the level of the meaning of the word itself. For example, if one says

"Glycosides lurk on every ghetto street corner."

it is probably because he has misunderstood the meaning of "glycoside".

It would seem that all three types of semigrammaticality are worthy of study, since all three occur in texts that a parsing system may be required to analyze. It does not seem possible to say that one type of semigrammaticality is more "serious" than another. A punctuation error causes a sentence to be less deviant than a selectional violation, a number error generally more so. Fragments can occur on every level.

In the next section an algorithm designed to yield information about semigrammatical sentences is given, and in the following section a classification of the restrictions into a hierarchy is suggested.

7.2. The Algorithm.

The semigrammaticality hierarchy may be set up in one of three ways. The user may employ the hierarchy specified in the system; this is explained in the next section. He may use the same classes of restrictions provided by the system (i.e. $PUNC, $SEL1, $WS, ...) but reorder them. Finally he may define his own classes of restrictions. The last is a good debugging device for he may specify those restrictions he suspects to be causing a grammatical sentence not to be accepted and lift them several at a time. In the SETL program

below the hierarchy is called sghier and is represented as an n-tuple

<sgtag1,sgtag2,...,sgtagk>.

As noted in Section 6.8 a slight modification is required in restrictiontest. Instead of returning t or f as the value of pas, restrictiontest returns a numerical value -- zero if it passes, the level of grammaticality if the restriction violated is in the hierarchy, and one more than the number of classes in the hierarchy otherwise. For example, if the hierarchy is

<$PUNC,$SEL1,$WS,$SS>

a grammatical construction would recieve a value of 0, and a construction with a punctuation error the value 1, a strong syntactic error the value 4, and a positional restriction violation 5. The spans are formed for grammatical constructions and those violating only restrictions in the hierarchy. Thus, in the algorithm of Section 6.8, in lines (3) and (5), the condition

pas le #sghier

means that any restrictions violated were in the hierarchy, and a new span is formed only if this condition holds.

A value given by the function sgval is associated with every span and indicates that span's level of grammaticality. In general, the sgval of a span is at least the minimum over all sets of spans in its divisions list of the maximum sgval

of spans in each set. More precisely, when a new span is formed from the concatenation of a first span and a second span, the sgval of the new span is the maximum of pas, the sgval of the first span, and the sgval of the second span (cf. line (5) of the algorithm in Section 6.8). A span cannot be more grammatical than its immediate descendants. When a new span is formed from a single span, its sgval is the maximum of pas and the original span's sgval (cf. line (3)). The sgval of a span for a category symbol is 0, since it must be grammatical; no restrictions apply at that point (cf. line (1)). When a new span is formed from an old span without any restrictions being applied, the sgval of the new span is simply that of the old (cf. line (6)). When two interpretations are merged under the same span, the sgval of this span is the minimum sgval of the two interpretations. All of this is illustrated in Figure 1.

Two interpretations merged under one span <p,A,q>

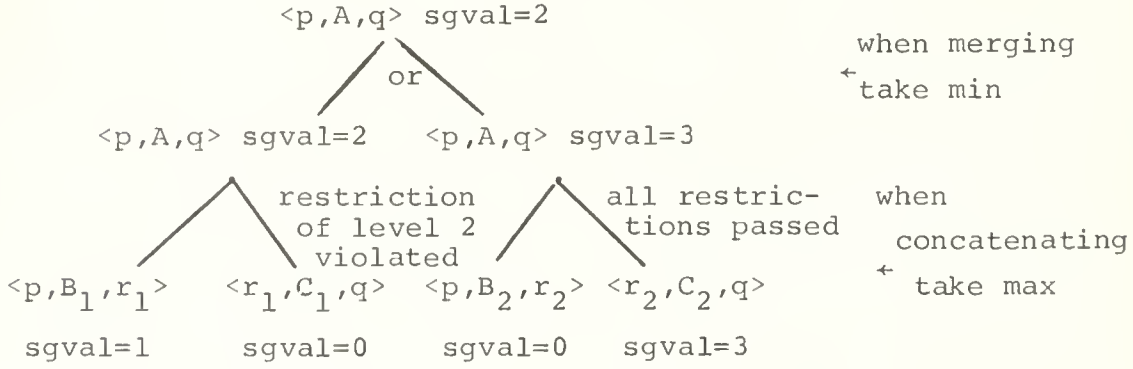


Figure 1

Once the parsing is completed and the user wants the results, he runs the subroutine readoff defined below. The results are put into the array parsetrees. If i is the highest level of grammaticality at which the entire sentence parses, parsetrees(i) gives that tree structure and success(i) = t. The function descend gives for each node all the sets of immediate descendents for that node on that level of grammaticality or higher. For the higher levels of grammaticality (i.e. smaller i) at which the entire sentence did not parse, the maximal well-formed "interesting" substrings are put into parsetrees together with the trees depending from them. "Interesting" is defined extensionally by listing the strings in the set strings. A possible set is suggested in the next section. Well-formed substrings which are spanned by larger well-formed substrings are not included in this listing unless they are descendents of other nodes.

spans is the collection of spans generated by the parser; pee extracts from a span its initial point, que its final point, and eta its symbol. sn is the number of words in the sentence. root is the root symbol of the grammar.

```
define  readoff;
/* the following are assumed to be global: parsetrees, descend,
     success, sghier, spans, pee, que, eta, sn, root,
     strings,  sgval, catnode, core */
(0 ≤ ∀level ≤ #sghier)
```

```
/* look for root symbols spanning entire sentence */
w = {x∈spans|<pee(x),que(x),eta(x),kay(x),jay(x)>
     eq <0,sn,root,0,0> and sgval(x) le level};  if w ne nl
/* find their descendants */
then success(level) = t;  (∀x∈w) down(x); end ∀x; return;
/* if none, find maximal interesting substrings and
     their descendants */
else ww = {z∈spans|eta(z)∈strings and <kay(z),jay(z)> eq<0,0>
         and sgval(z) le level};
   w = ww-{y|(∃z∈ww|(pee(y) gt pee(z) and que(y) le que(z)) or
                    (pee(y) ge pee(z) and que(y) lt que(z)))};
   (∀y∈w) down(y); end ∀y;;
end ∀level; return; end readoff;

define down(x);
/* add span x to parsetrees */
parsetrees(level) = parsetrees(level) with x;
if divlis(x) eq (catnode(core(x)) is xx) then descend(x)=xx;
     return;;
/* find valid descendants of x */
(∀y∈divlis(x)) if (∀z∈y|sgval(z) le level)
   then descend(x)=descend(x) with y;
      (∀zz∈y) down(zz); end ∀zz;;
end ∀y; return; end down;
```

7.3. The Hierarchy.

Well-Formed Substrings: When the parser cannot recognize a sentence in the input string at a given level of grammaticality, it reports the maximal well-formed substrings which are

"interesting" and their analyses.

The strings which are considered "interesting" by the present implementation are listed below. In general they are strings which are common or important enough to be given names in elementary grammars, strings which occur frequently in the grammar, and strings which are easily distinguished and open to little ambiguity. Special strings which appear only in the object position have not been included.

The center strings which can be part of larger strings: <ASSERTION>,<YESNOQ>.

The noun string <NSTG>, and strings of adjectives, <ASTG>, and adverbs, <DSTG>.

The most common or distinctive sentence nominalizations: <THATS>, <FORTOVO>.

The verb - object strings: <ASSERTIONVO> (the Verb Phrase), <VENO>, <VENPASS>, <VINGO>, <VO>, <TOVO>.

The prepositional phrase: <PN>.

Subordinate clauses beginning with subordinate conjunctions: <SUBO>, <SUB1>, <SUB2>, <SUB3>, <SUB4>, <SUB5>, <SUB6>, <SUB7>, <SUB8>.

Segments of the left adjunct of the noun: compound numbers <CPDNUMBER>, and the "how much of", "how many" constructions <HOWQSTG>, <HOWQASTG>.

The wh-strings: <WHETHS>, <WHETHTOVO>, <WHS-N>, <WHNS-N>, <PWHS-PN>, <PWHNS-PN>, <PWHS>, <PWHNS>, <WHATS-N>, <WHERES>, <WHENS>, <THATS-N>, <SAWHICHSTG>, <WHEVERS-N>.

Classification of Restrictions: The restrictions seem to fall into the following classes:

1. $RARE: These restrictions prohibit constructions which are not necessarily incorrect but occur infrequently. Examples of sentences rejected by $RARE restrictions include

"These are problems I've solved." (Normally a noun whose right adjunct is an assertion with a zeroed noun takes an article or quantifier in the left adjunct: "These are the problems I've solved.")

"He is all things to all people." (Normally the subject and object-of-be agree in number.)

$RARE is the only semigrammaticality class included in the grammar for the top-down parser, where it is activated by the RARE SWITCH.

In addition to $RARE restrictions there are $RARE constructions in the BNF component. For example, the underlined construction,

"John juggling three plates at once is a sight to behold."

rarely occurs in the subject position. These constructions are indicated by a minus sign just before the option in the grammar. A reasonable extension of the semigrammaticality hierarchy might be to define the other constructions in the BNF component for levels of grammaticality other than $RARE.

2. $PUNC: The punctuation restrictions include rules prohibiting or requiring commas in certain places, and

requiring a question mark at the end of a question. For example, some sentence adjuncts must be set apart from the rest of the sentence by commas, and commas are barred between a verb and its immediately following object. The following sentences would thus receive different analyses:

"The chemicals did not combine with the catalyst present."

"The chemicals did not combine, with the catalyst absent."

"He knows I know." / "He knows, I know."

The least a parsing system should have is a device for lifting punctuation restrictions since these are most susceptible to irregularities and unintentional errors on the part of author and printer.

3. $SEL1: The selectional restrictions include cooccurrence constraints between the verb and the subject and between the verb and the object. They are expressed by means of the attributes NOTNSUBJ and NOTNOBJ associated with the verb and listing attributes its subject and its object may not have. These restrictions not only operate between the main elements in a sentence but also between a noun and a present participle in its adjective position and between a gerund and a noun in its left adjunct.

There is enormouns room for expansion of the power of the grammar in this area. The restrictions required to check constraints are few and already in the grammar, but the lists of attributes under the NOTNSUBJ and NOTNOBJ nodes of most verbs in the lexicon are meager. For example, at

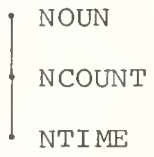
the present time few verbs ban inanimate subjects, a constraint needed to disambiguate

"Leaping fences can be dangerous."

A more significant extension would be possible if the tree-like structure of entires in the lexicon were more fully developed. For example, a selectional restriction requires a time noun in "a _____ ago", e.g. "an hour ago". Lifting this restriction allows the phrase "a grief ago", which has power as a metaphor. This metaphor may seem rather a result of the meaning of "grief", but in fact one can imagine contexts in which any noun which is possibly a count noun can fit reasonably well into the phrase:

"When did you start writing your dissertation?" "Three typewriters ago."

On the other hand, mass nouns fit much less well -- "cotton ago" -- and words other than nouns not at all -- "a the ago". If the attributes are structured as



constraints involving the lowest level attributes can be lifted first, then those for the next level, etc. This procedure would yield a richer hierarchy of semigrammatical sentences.[1] (It is for this possible expansion that we have

[1] This is substantially the program proposed by Noam Chomsky in "Some Methodological Remarks on Generative Grammar," Word 17 (1961), p. 219-239.

labelled the class $SEL1.)

Another possible extension involves sublanguages specific to particular fields. Certain very tight constraints hold in subfields of sciences which do not hold in the science as a whole nor in scientific English or general English as a whole. For example, in the subfield of pharmacology, the phrase

"changes in cells produced by drugs"

is unambiguous, for drugs can produce changes but not cells. On the other hand, one can imagine a paper somewhere in the larger field of biology which discusses drugs producing cells. One can construct a hierarchy of selectional restrictions corresponding to the level of the sublanguage. In addition to their increased disambiguating power these restrictions and the device for lifting them could be used in an information retrieval system to distinguish between sentences strictly in the sublanguage and sentences in scientific English as a whole, thus narrowing in on the field-specific information or hard facts of the scientific article.

4. $WS: The weak syntactic restrictions are few but include count noun and some case restrictions. The construction requiring "whom" rather than "who" before a wh-string with an omitted object or object-of-proposition is in this class. Thus, the sentence

"There is the man who I know you saw yesterday."

will not be recognized until the $WS class is lifted. The distinction between weak syntactic and strong syntactic

restrictions is somewhat arbitrary, but corresponds roughly to the intuitive feeling one might have about the seriousness of a violation.

This class could be broken into several levels in a manner similar to the selectional restrictions. For example, the phrase

"was injected into the muscle tissue of guinea pig, rat, and man"

is grammatical in the sublanguage of pharmacology, even though "guinea pig" and "rat" are count nouns in general English and require articles.

5. $SS: Among the strong syntactic restrictions are those requiring number agreement between the subject and verb, between article and quantifier, and between a noun and its appositive, that requiring the object of "between" and "among" to be nonsingular or aggregate, and that allowing a singular noun after a compound number only in the adjective position of the left adjunct of a noun ("twenty-five man life boat"). Most case restrictions are in this class as well, including those prohibiting such sentences as

"He wants you and I to see him."

"That is the boy whom I think broke the vase."

6. $POS5: This is a special classification of positional restrictions which should probably be included in the class $POS4. It contains cooccurrence constraints on the prepositions and adverbial prepositions which can occur in object strings following particular verbs. In

general, restrictions involving prepositions have been assigned to this class.

7. $POS4: Positional restrictions express structural properties of the grammar which are more easily and concisely stated as restrictions than as BNF productions. It is not envisioned that positional restrictions would be lifted, except to test the relative power of different parts of the grammar, since it would open the door to nonsense strings and nonsense parses of good sentences. The $POS4 class contains the positional restrictions which refer to the literals and attributes of lexical items. By far the majority of restrictions fall into this class and these restrictions do the bulk of the work of the grammar.

8. $POS3: Restrictions in this class do not refer to attributes of words but only to their categories or else to symbols in the BNF component. An example would be a restriction governing the option an adjunct might be allowed to take given the value of its host.

9. $POS2: This class of restrictions deals with the occurrence of null nodes in constructions, in particular restrictions governing zeroed nouns in wh-strings,

"the man who () wouldn't leave",

and noun strings with null noun variants,

"Give me those three ()."

10. $POS1: This class is intended to include any restriction whose sole purpose is to block a redundant

analysis the BNF component would otherwise produce. There are no restrictions in this class at the present time and it is hoped they will be avoided.

The order in which these classes are listed seems to be the most reasonable ordering of the hierarchy, with the classes $RARE, $PUNC, $SEL1, $WS, $SS, being successively lifted. The user may override this with his own ordering if he wishes or he may define his own classes extensionally. Of course the more classes of restrictions one wants to have lifted, the more the spans that will be generated, bringing one closer to the situation described in Chapter 4.

BIBLIOGRAPHY

1. Bloomfield, Leonard, Language, New York, 1933.
2. Bross, I.D.J., Shapiro, P.A., and Anderson, B.B., "How Information is Carried in Scientific Sublanguages," Science, June 1972, p. 1303-1309.
3. Cheatham, T.E., and Sattley, K., "Syntax-Directed Compiling," Proceedings of Eastern Joint Computer Conference, AFIPS, Vol. 25, 1964, pp. 31-57.
4. Chomsky, Noam, Syntactic Structures, The Hague, 1957.
5. _____, "On Certain Formal Properties of Grammars," Information and Control, Vol. 2, 1959, pp. 137-167.
6. _____, "Some Methodological Remarks on Generative Grammar," Word 17, 1961, pp. 219-239.
7. _____, Aspects of the Theory of Syntax, Cambridge Mass., 1965.
8. Cocke, John, and Schwartz, J.T., Compilers and Compiler Writing Techniques, New York University, 1969
9. Earley, Jay, An Efficient Context-Free Parsing Algorithm, Ph.D. Thesis, Carnegie-Mellon University, 1968.
10. Floyd, Robert W., "Syntactic Analysis and Operator Precedence," Journal of the ACM, Vol. 10, July 1963. pp. 316-333.
11. _____, "The Syntax of Programming Languages -- A Survey," IEEE Transactions on Electronic Computers, Vol. EC-13, August 1964, pp. 346-353.
12. Gleason, H.A., Linguistics and English Grammar, New York 1965.

13. Grishman, Ralph, "The Implementation of the String Parser," in Courant Computer Science Symposium 8: Natural Language Processing, R. Rustin, ed., New Y!rk, 1973.

14. Harris, Zellig, "From Morpheme to Utterance," Language, Vol. 22, 1946, pp. 161-183.

15. _____, "Discourse Analysis," Language, Vol. 28, 1952, pp. 1-30.

16. _____, String Analysis of Sentence Structure, The Hague, 1962.

17. Hays, David G., "Automatic Language-Data Processing," in Computer Applications in the Behavioral Sciences, Harold Borko, ed., Englewood Cliffs, N.J., 1962, pp. 411-414.

18. Hopcroft, John E., and Ullman, Jeffrey D., Formal Languages and Their Relation to Automata, Reading, Mass., 1969.

19. Irons, E.T., "A Syntax-Directed Compiler for ALGOL 60", Communications of the ACM, Vol. 4, January 1961, pp. 51-55.

20. Kasami, T., and Torii, K., "A Syntax-Analysis Procedure for Unambiguous Context-Free Grammars," Journal of the ACM, Vol. 16, July 1969, pp. 423-431.

21. Knobe, Bruce, An Experimental Comparison of the Efficiency of Parsing Algorithms, Ph.D. Thesis, New York University, June 1972.

22. Kuno, Susumu, and Oettinger, Anthony G., "Syntactic Structure and Ambiguity of English," Proceedings of Fall Joint Computer Conference, 1963, pp. 397-418.

23. Naur, Peter, ed., "Report on the Algorithmic Language ALGOL 60," Communications of the ACM, Vol. 3, 1960, p. 299.

24. Robinson, Jane, and Marks, Shriley, PARSE: A System for Automatic Syntactic Analysis of English Text, Rand Corp., 1965.

25. Sager, Naomi, String Program Report No. 4: A Computer String-Grammar of English, Parts I, II, New York, 1968.

26. _____, "A Hierarchy of BNF Grammars for Parsing English: A Sublanguage Approach," talk delivered at University of California, Berkeley, Department of Electrical Engineering and Computer Sciences, April 14, 1971.

27. _____, "Syntactic Formatting of Scientific Information," AFIPS Conference Proceedings, Vol. 41, 1972, p. 791.

28. _____, "A Two-Stage BNF Specification of Natural Language," Journal of Cybernetics, Vol. 2, 1972, pp. 39-50.

29. Schwartz, J. T., "The Analysis of Natural Language," unpublished manuscript, undated.

30. _____, "Generalized Nodal Spans Parse Routine, Preliminary Draft," SETL Newsletter #46, 1971.

31. Wells, Rulon S., "Immediate Constituents," Language, Vol. 23, 1947, pp. 81-117.

32. Younger, Daniel H., "Recognition and Parsing of Context-Free Languages in Time n^3," Information and Control, Vol. 10, February 1967, pp. 189-208.

33. Zwicky, Arnold, Friedman, Joyce, Hall, Barbara C., and Walker, Donald, "The MITRE Syntactic Analysis Procedure for Transformational Grammars," Proceedings of Fall Joint Computer Conference, 1965, pp. 317-326.

APPENDIX A

STRING GRAMMAR OF ENGLISH FOR BOTTOM-UP PARSING

PRODUCTIONS WHOSE RIGHT SIDES ARE STRINGS OF SYMBOLS ARE FOLLOWED BY EXAMPLES.

SYMBOLS OF THE GRAMMAR ARE NAMES ENCLOSED IN TRIANGULAR BRACKETS: <ASSERTION>, <*V>; WORDS WITHOUT THE BRACKETS ARE ENGLISH WORDS: AND, THAT, WHO. NAMES BEGINNING WITH "*" ARE TERMINAL SYMBOLS OR "PARTS OF SPEECH": <*N>, <*V>. OTHERS ARE NONTERMINAL SYMBOLS: <ASSERTION>, <NSTG>.

IN THE NAMES OF THE SYMBOLS, THE FOLLOWING MNEMONIC SYSTEM IS USED:

- N = NOUN
- NS = POSSESSIVE NOUN
- PRO = PRONOUN
- W = TENSE WORD
- V = UNTENSED VERB
- TV = TENSED VERB
- VING = PRESENT PARTICIPLE OF VERB
- VEN = PAST PARTICIPLE OF A VERB
- A, ADJ = ADJECTIVE
- D = ADVERB
- Q = QUANTIFIER
- T = ARTICLE
- P = PREPOSITION
- DP = ADVERBIAL PREPOSITION
- CS0 - CS7 = VARIOUS CLASSES OF SUBORDINATING CONJUNCTIONS
- WH = WH-WORD: WHO, WHICH, ETC.
- INT = INTERJECTION

THESE MAY BE CONCATENATED. THUS "<PN>" MEANS A PREPOSITION FOLLOWED BY A NOUN, I.E. A PREPOSITIONAL PHRASE, AND "<NA>" MEANS A NOUN FOLLOWED BY AN ADJECTIVE (IN THE OBJECT POSITION). "STG" STANDS FOR "STRING" SO "<NSTG>" MEANS NOUN STRING AND "<DSTG>" MEANS A STRING OF ADVERBS. "VAR" STANDS FOR "VARIANT" SO "<NVAR>" MEANS THE POSSIBLE VARIANTS OF OF A NOUN. "POS" STANDS FOR "POSITION" SO "<TPOS>" MEANS THE ARTICLE POSITION IN A NOUN PHRASE.

ASSERTIONS ARE SOMETIMES REPRESENTED BY "S" AND QUESTIONS BY "Q". THUS "<THATS>" MEANS THE WORD "THAT" FOLLOWED BY AN ASSERTION. SUBJECT IS REPRESENTED BY "S" OR "SUBJ" AND OBJECT BY "O" OR "OBJ". EXAMPLES:

- <OBJBE> = OBJECT OF THE VERB "TO BE"
- <TOVO> = INFINITIVE CONSTRUCTION: THE WORD "TO" FOLLOWED BY A VERB FOLLOWED BY AN OBJECT.
- <SASOBJBE> = SUBJECT = THE WORD "AS" = OBJECT OF THE VERB "TO BE"

"L" STANDS FOR "LEFT" AND "R" FOR "RIGHT". EXAMPLES:

- <LN> = LEFT ADJUNCT OF A NOUN
- <RV> = RIGHT ADJUNCT OF A VERB
- <LNR> = A NOUN WITH ITS LEFT AND RIGHT ADJUNCTS, I.E. A NOUN PHRASE
- <RSUBJ> = ADJUNCT OF THE SUBJECT PLACED TO THE RIGHT OF THE OBJECT.

"<SA>" STANDS FOR "SENTENCE ADJUNCT".

```
*       THE MINUS SIGN "-" INDICATES THAT WHAT FOLLOWS HAS BEEN DELETED FROM
*  THE SUBJECT OR OBJECT POSITION OF THE ASSERTION OR YES-NO QUESTION IN THE
*  STRING.  THUS
*          <WHNS-N>  =  WH-WORD = NOUN = ASERTION WITH NOUN DELETED FROM
*               SUBJECT OR OBJECT POSITION.
*       THE PREFIX "REP"  IS USED IN SOME RECURSIVE PRODUCTIONS.  THE PREFIX
*  "NUL" IS USED IN PRODUCTIONS WHOSE ONLY INTENT IS TO SIMPLIFY ANOTHER
*  PRODUCTION.
*       AN ALPHABETICAL LISTING OF THE NONTERMINAL SYMBOLS TOGETHER WITH
*  THE NUMBER OF THE CORRESPONDING PRODUCTION APPEARS AT THE END OF THE
*  BNF GRAMMAR.
*       ALTERNATIVES ARE SEPARATED BY A SLASH "/".  SQUARE BRACKETS "[" "]"
*  INDICATE THAT WHAT IS ENCLOSED IS OPTIONAL.  A HYPHEN "-" BEFORE A SYMBOL
*  INDICATES THAT THE ALTERNATIVE IS RARE.
*       A "C" FOLLOWS THE SYMBOL FOR THE CORE ELEMENT OF A STRING.  AS
*  MATCHING WITH THE STRING PROCEEDS, THE FIRST ELEMENT IS TAKEN AS CORE
*  ELEMENT UNTIL THE FIRST SYMBOL FOLLOWED BY A "C" IS ENCOUNTERED.  THEN THAT
*  ELEMENT IS TAKEN AS CORE ELEMENT UNTIL THE NEXT SUCH ELEMENT IS ENCOUNTERED.
*
*
* 1. SENTENCE
*
<SENTENCE>        ::= [<INTRODUCER>] <CENTER>C <ENDMARK> .                  1.1
*                     BUT THIS IS NOT ALWAYS POSSIBLE.
<INTRODUCER>      ::= AND / OR / BUT / FOR .                                1.2
<CENTER>          ::= <ASSERTION> / <QUESTION> /<PROSENT>/<PERMUTATION>.    1.3
<ENDMARK>         ::= '.' / '?' / ';'  .                                    1.4
*
* 2. CENTER STRINGS
*
<ASSERTION>       ::= [<SA>] [<SUBJECT>]C [<SA>] <ASSERTIONVO>C .           2.1
<ASSERTIONVO>     ::= <VERB> [<SA>] [<OBJECT>] [<RV>] [<SA>] .              2.1A
*                     HE HIT THE BALL SOLIDLY FOR THE FIRST TIME TODAY
<QUESTION>        ::= <YESNOQ> / <WHQ-N> / <WHQ> / <PWHQ-PN> /              2.2
                      <PWHQ> / <WHNQ-N> / <PWHNQ-PN> / <PWHNQ>  .
<PROSENT>         ::= THIS <SA>C .                                          2.3
*                     THIS THOUGH EVERY SPEAKER WILL ACCEPT UTTERANCES THAT HE
*                          HAS NOT HEARD BEFORE
<PERMUTATION>     ::= <TSUBJVO> / <OBES> .                                  2.4
*
* 3.  QUESTION CENTERS
*
<YESNOQ>          ::= [<SA>] <VERB1>C <SUBJECT> [<RW>] [<SA>]               3.1
                      [<YESNOQVO>]C .
<YESNOQVO>        ::= [<VERB2>] [<OBJECT>] [<RV>] [<SA>] [<ORNOT>] .        3.1A
*                     DID HE HIT THE BALL SOLIDLY OR NOT
<ORNOT>           ::= OR NOT .                                              3.2
<WHQ-N>           ::= <NUL1WHQ-N> <NUL2WHQ-N>C .                            3.3
*                     WHAT DID HE HIT SOLIDLY
<NUL1WHQ-N>       ::= WHO / WHOM / WHICH / WHAT / WHOSE .                   3.3A
<NUL2WHQ-N>       ::= <YESNOQ> / <ASSERTION> .                              3.3B
<WHQ>             ::= [<SA>] <NULWHQ> <YESNOQ> .                            3.4
*                     HOW DID HE HIT THE BALL
```

```
<NULWHQ>          ::= WHERE / WHEN / HOW / WHY .                              3.4A
<PWHQ-PN>         ::= [<SA>] <*P>C <NULPWHQ> <YESNOQ>C .                      3.5
*                     ON WHOM CAN WE RELY
<PWHQ>            ::= [<SA>] <*P>C <NULPWHQ> <YESNOQ>C .                      3.6
*                     WITH WHAT DID HE HIT THE BALL
<NULPWHQ>         ::= WHOM / WHICH / WHAT / WHOSE .                           3.6A
<WHNQ-N>          ::= [<SA>] <WHN>C <NUL2WHQ-N>C .                            3.7
*                     WHICH BALL DID HE HIT SOLIDLY
<PWHNQ-PN>        ::= [<SA>] <*P>C <WHN> <YESNOQ>C .                          3.8
*                     ON WHICH PLAYERS CAN WE DEPEND
<PWHNQ>           ::= [<SA>] <*P>C <WHN> <YESNCQ>C .                          3.9
*                     WITH WHICH BAT DID HE HIT THE BALL
<WHN>             ::= <LNR> / <VINGOFN> .                                     3.9A
*
* 4.  PERMUTATION CENTERS
*
<TSUBJVO>         ::= <NULTSUBJVO> [<SA>] <VERB1>C <SUBJECT> [<SA>]           4.1
                      [<TSUBJVCVO>]C .
<NULTSUBJVO>      ::= NEITHER / NOR / <DSTG> / <PN> .                         4.1A
<TSUBJVOVC>       ::= [<VERB2>] [<OBJECT>] [<RV>] [<SA>] .                    4.1B
*                     NEITHER WILL I BE INTIMIDATED
<CBES>            ::= <NULCBES> [<SA>] <VERB3>C [<VERB4>]C [<SA>]             4.2
                      <SUBJECT> [<SA>] .
<NULCBES>         ::= <PN> / <ASTG> / <VENPASS> / <DSTG> .                    4.2A
*                     IN ITALY ARE MANY BEAUTIFUL CHURCHES
*
* 5.  SUBJECT STRINGS
*
<SUBJECT>         ::= <NSTG> / <VINGSTG> / <SN> / THERE .                     5.1
<NSTG>            ::= <LNR> / <NWHSTG> / <NAMESTG> [<RN>] / <LPRCR>.          5.2
<LNR>             ::= <LNNVAR> [<RN>] / <LN1> [<RN>] .                        5.3
*                     THE THREE NEW HOUSE BOATS IN THE HARBOR / THE THREE
*                          NEWEST IN THE HARBOR
<LNNVAR>          ::= [<TQPOS>] <ANNVAR>C .                                   5.3A
*                     THE THREE NEW HOUSE BOATS
<TQPOS>           ::= [<TPOS>] [<QPOS>] .                                     5.3B
*                     THE THREE
<ANNVAR>          ::= [<APOS>] <NNVAR>C .                                     5.3C
*                     NEW HOUSE BOATS
<NNVAR>           ::= [<NPOS>] <NVAR>C .                                      5.3D
*                     HOUSE BOATS
<LN1>             ::= [<TPOS>] [<QPOS>] [<APOS>]                              5.3E
*                     THE THREE NEWEST
<NVAR>            ::= <*N> / <*VING> .                                        5.4
<NWHSTG>          ::= <WHATS-N> / <WHERES> / <WHEVERS-N> .                    5.5
<NAMESTG>         ::= <*N> <NULNAMESTG> / <NAMESTG> <NULNAMESTG> .            5.6
*                     GEORGE BERNARD SHAW
<NULNAMESTG>      ::= <NUMBRSTG> / <*N> .                                     5.6A
<LPRCR>           ::= [<LPRO>] <*PRO>C [<RN>] .                               5.7
*                     ONLY YOU
<VINGSTG>         ::= <NSVINGO> / <VINGOFN> .                                 5.8
<NSVINGO>         ::= [<TPOS>] <VINGO>C .                                     5.9
*                     HIS HITTING THE BALL (SURPRISED US)
```

```
<VINGOFN>       ::= [<LN2>] <*VING>C [OF] [<OBJECT>] [<RV>] [<SA>] .        5.10
*                   THE LIFTING OF THE BLOCKADE (SURPRISED EVERYONE)
<LN2>           ::= [<TPOS>] [<GPOS>] [<APOS>]                           5.10A
<SN>            ::= <THATS> / <FORTOVO> / <TOVO> / - <SVINGO> /           5.11
                    <CISHOULD> / <SNWH> / - <THATVING> .
*
* 6.  SENTENCE NOMINALIZATION STRINGS SN
*
<THATS>         ::= THAT <ASSERTION>C .                                   6.1
*                   THAT THIS LEMMA IS TRUE (IS OBVIOUS)
<FORTOVO>       ::= FOR <SUBJECT>C [<SA>] <TOVO>C .                       6.2
*                   FOR YOU TO PROVE THIS LEMMA (IS TOO MUCH TO EXPECT)
<SVINGO>        ::= <NULSVINGO> <VINGO>C .                                6.3
*                   (OBSERVE) THE STUDENT BECOMING MORE INTENT
<NULSVINGO>     ::= <NSTG> / <VINGSTG> .                                  6.3A
<THATVING>      ::= <SN> <VINGO>C .                                       6.4
*                   FOR YOU TO PROVE THIS LEMMA BEING CONSIDERED UNLIKELY
*                   (SHOULD NOT INSULT YOU)
<CISHOULD>      ::= [THAT] <ASSERTION>C .                                 6.5
*                   THAT WE PROVE THIS LEMMA (IS IMPERATIVE)
<SNWH>          ::= <WHETHS> / <WHETHTOVO> / <WHS-N> / <PWHS> /           6.6
                    <PWHS-PN> / <WHNS-N> / <PWHNS> / <PWHNS-PN> .
*
* 7.  VERB AND VERBAL OBJECT STRINGS
*
<VERB>          ::= <LTVR> / <LWR> [<SA>] <LVR>C / <LVR> .                7.1
*                   BAKES / WILL UNDOUBTEDLY BAKE / (HAVE HER) BAKE (A CAKE)
<LTVR>          ::= [<LV>] <*TV>C [<RW>] [<RV>] .                         7.2
*                   NEVER COCKS QUICKLY
<LWR>           ::= [<LW>] <*W>C [<RW>] .                                 7.3
*                   NEVER WILL
<LVR>           ::= [<LV>] <*V>C [<RV>] .                                 7.4
<VENO>          ::= <LVENR> [<SA>] [<OBJECT>] [<RV>]  .                   7.5
*                   BAKED A CAKE
<LVENR>         ::= [<LV>] <*VEN>C [<RV>] .                               7.6
<VENPASS>       ::= [<LVSA>] <LVENR>C [<SA>] [<PASSOBJ>] [<RV>]           7.7
                    [<SA>] .
*                   (I WAS) GIVEN A CAKE BY HER
<LVSA>          ::= <DSTG>  .                                             7.8
<VINGO>         ::= [<LVSA>] <LVINGR>C [<SA>] [<OBJECT>] [<RV>]           7.9
                    [<SA>] .
*                   BAKING A CAKE
<LVINGR>        ::= [<LV>] <*VING>C [<RV>] .                              7.10
<VO>            ::= <LVR> [<SA>] [<OBJECT>] [<RV>] .                      7.11
                  BAKE A CAKE
<TOVO>          ::= [<LV>] TO <LVR>C [<SA>] [<OBJECT>] [<RV>] [<SA>].     7.12
*                   TO BAKE A CAKE
<VERB1>         ::= <LWR> / <LTVR> .                                      7.13
<VERB2>         ::= <LVR> [<SA>] .                                        7.14
```

```
<VERB3>          ::= <LTVR> / <**W> [<RW>] [<SA>] <LVR>C .                    7.15
<VERB4>          ::= <*VEN> / TO <*V>C .                                      7.16
*
* 8.  OBJECT STRINGS
*
<OBJECT>         ::= <THATS> / <NSTGO> / <TOVO> / <PN> / <PVINGSTG> /         8.1
                     <PVINGO> / <PSNWH> / <PNN> / <PNVINGSTG> / <PSTG> /
                     <PNTHATS> / <PNSNWH> / <PNTHATSVO> / <PSVINGO> /
                     <DP2> / <DP4> / <DP2PN> / <DP4PN> / <DPSN> /
                     <DP1PN> / <DP1> / <OBJECTBE>  / <VENO> /
                     <VO>  /   <SASOBJBE> / <NPN> /
                     <VINGSTGPN> / <NPVINGSTG> / <ASOBJBE> / <NPSVINGO>
                     / <NPVINGO> / <NPSNWH> / <DP3> / <DP3PN> /
                     <OBJBE> / <VINGSTG> / <SNWH> / <WHETHS> /
                     <SVINGO> / <C1SHOULD> / <FORTOVO> / <NTOVO> /
                     <NN> / <SOBJBE> / <SVEN> / <NA> / <ADJN> / <ND>
                     / <VINGO> / <SVO> / <ASTG> / <NTHATS>
                     / <DSTG> / <NSNWH> / <ASSERTION> / <STOVO=N> .
<PASSOBJ>        ::= <THATS> / <PN> / <PVINGSTG> / <PSVINGO> /               8.2
                     <PVINGO> / <PSNWH> / <ASOBJBE> / <PSTG> /
                     <DP1> / <P1> / <DP1P> / <DP1PN> /
                     <VINGO> / <FORTOVO> / <TOVO> / <ASSERTION> /
                     <NSTGO> / <SNWH> / <OBJBE> / <VENPASS> / <ASTG>
                     / <DSTG> / <VO> .
<OBJECTBE>       ::= <VENPASS> / <VINGO> / <OBJBE> / <SN> / = <VINGSTG>     8.3
                     / <TOVO> / <ASSERTION> / <BEINGO> .
<OBJBE>          ::= <ASTG> / <NSTG> / <PN> / <DSTG> .                       8.4
<BEINGO>         ::= [<LV>] BEING <NULBEINGO>C .                             8.5
<NULBEINGO>      ::= <LAR> / <NSTG> .                                        8.5A
*                    (IS) BEING FOOLISH / (IS) BEING A FOOL
<NSTGO>          ::= <NSTG> .                                                8.6
<NN>             ::= <NSTGO> [<SA>] [<NSTGO>]C .                             8.7
*                    (BUY) HIM A BOOK
<ASTG>           ::= <LAR> / <LONR> .                                        8.8
*                    (IS) LONG / (IS) 300 PAGES LONG
<LAR>            ::= [<LA>] <AVAR>C [<RA>] .                                 8.9
*
* 8A. P STRINGS
*
<P1>             ::= <*P> .                                                  8A.1
*                    (YOU ARE DISAPPROVED) OF
<PSTG>           ::= <PN> / <PVINGSTG> / <PSVINGO> .                         8A.2
<PN>             ::= [<LP>] <*P>C [<NSTGO>] .                                8A.3
*                    (I DISAPPROVE) OF YOU; (LEAVE) IN HASTE (RV); IN
*                    FACT (SA)
*  NOTE:  PREPOSITION IS CORE ELEMENT IN MOST STRINGS INVOLVING PREPOSITIONS.
<PVINGSTG>       ::= <*P> <VINGSTG> .                                        8A.4
*                    (RELY) ON THEIR SENDING THE MONEY QUICKLY; (RELY) ON
*                    THEIR QUICK SENDING OF THE MONEY
<PSVINGO>        ::= <*P> <SVINGO> .                                         8A.5
*                    (ACCOUNT) FOR THE MONEY'S ABSENCE NOT BEING NOTICED
<PVINGO>         ::= <*P> <VINGO> .                                          8A.6
```

```
*                    (SUCCEED) IN CONFUSING THE ISSUE
<PSNWH>          ::= <*P> <SNWH> .                                        8A.7
*                    (INQUIRE) ABOUT WHETHER THEY ARE HERE
<NPN>            ::= [<NSTGO>] [<SA>] [<PN>]C .                           8A.8
*                    (ATTRIBUTE) THEIR DIFFICULTIES TO MISMANAGEMENT
<PNN>            ::= <PN> [<SA>] [<NSTGO>] .                              8A.9
*                    (ATTRIBUTE) TO MISMANAGEMENT THEIR GRAVE DIFFICULTIES
<VINGSTGPN>      ::= <VINGSTG> [<SA>] <NULVINGSTGPN>C .                   8A.10
<NULVINGSTGPN>   ::= <PN> / <PVINGSTG> / <PSVINGO> / <PVINGO> .           8A.10A
*                    (LEAVE) CANVASSING TO HIS AIDES; (ATTRIBUTE) HIS
*                    LEAVING TO HIS BEING OVERWORKED
<PNVINGSTG>      ::= <NULVINGSTGPN> [<SA>] <VINGSTG>C .                   8A.11
*                    (LEAVE) TO HIS SECRETARY MAKING ALL ARRANGEMENTS
<NPVINGSTG>      ::= [<NSTGO>] [<SA>] <PVINGSTG>C .                       8A.12
*                    (ASK) HIM ABOUT THEIR OFFERING HIM MORE MONEY
<NPSVINGO>       ::= [<NSTGO>] [<SA>] <PSVINGO>C .                        8A.13
*                    (ASK) HIM ABOUT NO ONE HAVING BEEN THERE
<NPVINGO>        ::= [<NSTGO>] [<SA>] <PVINGO>C .                         8A.14
*                    (PREVENT) HIM FROM RUINING HIS HEALTH
<NPSNWH>         ::= [<NSTGO>] [<SA>] <*P> <SNWH> .                       8A.15
*                    (ASK) HIM ABOUT WHETHER THE PASSPORTS WILL BE STAMPED
*
*
* 8B.  DP STRINGS
*
<DP1>            ::= <*DP> .                                              8B.1
*                    (CARRY) ON
<DP2>            ::= <*DP> [<NULNV>] .                                    8B.2
*                    (LOOK) UP THE NUMBER
<DP3>            ::= [<NULNV>] <*DP>C .                                   8B.3
*                    (LOOK) THE NUMBER UP
<DP4>            ::= <*DP> OF [<NULNV>] .                                 8B.4
*                    (LET) OUT THAT A CHANGE IN GOVERNMENT WOULD BE WELCOME
<NULNV>          ::= <NSTGO> / <VINGSTG> .                                8B.4A
<DPSN>           ::= <*DP> [<SA>] <SN> .                                  8B.5
*                    (FIND) OUT WHETHER HE IS COMING
<DP1PN>          ::= <*DP> [<SA>] [<NULDPPN>]C .                          8B.6
<NULDPPN>        ::= <PN> / <PVINGSTG> / <PSVINGO> .                      8B.6A
*                    (BOILS) DOWN TO HIS TAKING A BRIBE
<DP2PN>          ::= <DP2> <NULDPPN>C .                                   8B.7
*                    (MIX) UP THE LAST NAME WITH THE FIRST
<DP3PN>          ::= <DP3> <NULDPPN>C .                                   8B.8
*                    (MIX) IT UP WITH SOMETHING ELSE
<DP4PN>          ::= <DP4> <NULDPPN>C .                                   8B.9
*                    (MIXING) UP OF THE LAST NAME WITH THE FIRST
<DP1P>           ::= <*DP> <*P>C .                                        8B.10
*                    (WAS LOOKED) UP TO
*
*
* 8C.  OBJECT NOMINALIZATION STRINGS
*
<NTOVC>          ::= [<SUBJECT>] [<RV>] [<SA>] <TOVO>C .                  8C.1
*                    (CAUSE) ME TO MAKE AN ERROR
<SVO>            ::= <ASSERTION> .                                        8C.2
*                    (MAKE) ME FAIL
```

```
<NTHATS>          ::= [<NSTGO>] [<RV>] [<SA>] <NULNTHATS>C .                    8C.3
<NULNTHATS>       ::= <THATS> / <ASSERTION> .                                   8C.3A
*                     (TELL) ME THAT I AM WRONG
<NSNWH>           ::= [<NSTGO>] [<RV>] [<SA>] <SNWH>C .                         8C.4
*                     (ASKED) ME WHETHER HE WAS RIGHT
<PNTHATS>         ::= <*P> [<NSTGO>] [<RV>] [<SA>] <NULNTHATS> .                8C.5
*                     (REPORT) TO HIM THAT HE WAS WRONG
<PNTHATSVO>       ::= <PN> [<RV>] [<SA>] <O1SHOULD> .                           8C.6
*                     (REQUIRE) OF HIM  THAT HE MAKE NO ERRORS
<PNSNWH>          ::= <PN> [<RV>] [<SA>] <SNWH> .                               8C.7
*                     (INQUIRED) OF ME WHETHER HE WAS WRONG
<STOVC=N>         ::= <SUBJECT> [<SA>] <TOVO=N>C .                              8C.8
*                     (HAS) HIS FAMILY TO CONSIDER
*
*
* 8D. NOMINALIZATIONS WITH ZEROED VERB BE
*
<SOBJBE>          ::= [<SUBJECT>] [<SA>] [<OBJBE>]C [<SA>] .                    8D.1
*                     (ELECT) HIM PRESIDENT
<SVEN>            ::= [<SUBJECT>] [<SA>] <VENPASS>C .                           8D.2
*                     (HAVE) THE SUSPECTS FINGERPRINTED
<NA>              ::= [<NSTGO>] [<SA>] <LAR>C .                                 8D.3
*                     (MAKE) IT AVAILABLE
<ADJN>            ::= <ASTG> [<SA>] [<NSTGO>]C .                                8D.4
*                     (MAKE) AVAILABLE A PRINTED FORM
<ND>              ::= [<NSTGO>] [<SA>] <DSTG>C .                                8D.5
*                     (LEAVE) THE DOOR AJAR
<SASCBJBE>        ::= [<SUBJECT>] [<SA>] AS [<NULSASOBJBE>]C [<SA>] .           8D.6
<NULSASOBJBE>     ::= <OBJBE> [<SA>] / = <VINGO> .                              8D.6A
*                     (IDENTIFY) THE PAINTING AS A GENUINE VERMEER
<ASOBJBE>         ::= AS [<OBJBE>] .                                            8D.7
*                     (SERVE) AS CONTROLS
*
* 9.  SENTENCE ADJUNCT STRINGS
*
<SA>              ::= <REPSA> [<SA>]  .                                         9.1
<REPSA>           ::= <*INT>/.<DSTG> [<RD>] / <PN> / <PA> / <NSTGT>/            9.1A
                      <RSUBJ> / <RNSUBJ> / [<LCS>] <CSSTG>C / <OBJBESA> /
                      <SOBJBESA> / <VINGO> / <VENPASS> / <SAWH> /
                      <TOVC> / <NVSA> .
<DSTG>            ::= <*D> / <*D> [<DSTG>]C .                                   9.2
*                     ODDLY ENOUGH
<PA>              ::= <*P> <*ADJ> .                                             9.3
*                     IN GENERAL
<NSTGT>           ::= <LNR> .                                                   9.4
*                     TEN YEARS AGO
<RSUBJ>           ::= <NUL1RSUBJ> <NUL2RSUBJ>C / <*PRO> .                       9.5
<NUL1RSUBJ>       ::= <*Q> / <*T> .                                             9.5A
<NUL2RSUBJ>       ::= <PN> / <*D> .                                             9.5B
*                     ALL OF THEM; HIMSELF
<RNSUBJ>          ::= <SN> / <RNWH> / = <THATS=N> .                             9.6
*                     (THE HYPOTHESIS WAS PROPOSED) THAT THE EARTH IS ROUND
<CSSTG>           ::= <SUB1> / <SUB2> / <SUB3 > / <SUB0> / <SUB5> /             9.7
                      <SUB6> / <SUB4> / <SUB8> .
```

```
<OBJBESA>          ::= <OBJBE> [<SA>] .                                          9.8
*                      AN INVETERATE GAMBLER, (HE,..)
<SOBJBESA>         ::= <NSTG> [<SA>] [<NULSOBJBESA>]C .                          9.9
<NULSCBJBESA>      ::= <OBJBESA> / <VENPASS> / <VINGO> .                         9.9A
*                      THE WAR HAVING ENDED, (PEOPLE BEGAN TO,..)
<SAWH>             ::= <SAWHICHSTG> / <WHETHS> / <WHEVERS-N> .                   9.10
<NVSA>             ::= <NSTG> [<SA>] <VERB>C .                                   9.11
*                      (HE WILL,) IT SEEMS, (BE THERE ALONE)
*
* 10.  SUBORDINATE CONJUNCTION (CS) STRINGS
*
<SUB0>             ::= <*CS0> <OBJBE>C [<SA>] .                                  10.1
*                      THOUGH BLIND
<SUB1>             ::= <*CS1> <ASSERTION>C .                                     10.2
*                      BECAUSE HE HAD INTIMATE KNOWLEDGE OF THE UNDERWORLD
<SUB2>             ::= <NULSUB2> <VENPASS>C .                                    10.3
<NULSUB2>          ::= <*CS2> / AS / THAN .                                      10.3A
*                      ALTHOUGH CAPTURED BY THE SYNDICATE
<SUB3>             ::= <*CS3> <VINGO>C .                                         10.4
*                      BEFORE DROPPING HIS GUN
<SUB4>             ::= <*CS4> <VINGSTG>C .                                       10.5
*                      AFTER HIS REFUSING TO PLEAD FOR HIS LIFE
<SUB5>             ::= <*CS5> <SVINGO>C .                                        10.6
*                      WITHOUT ANYONE TRYING TO SAVE HIM
<SUB6>             ::= <*CS6> <SOBJBE>C .                                        10.7
*                      WITH HIS GUN OUT OF REACH
<SUB7>             ::= <*CS7> <SVEN>C .                                          10.8
*                      WITH THE MYSTERY STILL UNSOLVED
<SUB8>             ::= <NULSUB8> <*D>C .                                         10.9
<NULSUB8>          ::= AS / THAN .                                               10.9A
*                      AS ALWAYS
*
* 11.  RN RIGHT ADJUNCTS OF N
*
<RN>               ::= <REPRN> [<RN>] .                                          11.1
<REPRN>            ::= <VENPASS> / <RNP> / <RNWH> / <THATS-N> / <S-N> /          11.1A
                       <SN> / <VINGO> / <TOVO> / <TOVO-N> / <FORTOVO-N>/
                       <ADJINRN> / <DSTG> / <APPOS> / WHATSOEVER .
<RNP>              ::= <PN> / <PVINGSTG> / <PSVINGO> / <PSNWH> .                 11.2
<RNWH>             ::= <WHS-N> / <WHENS> / <PWHS> / <PWHS-PN> /                  11.3
                       <WHNS-N>   / <PWHNS-PN> / - <PWHNS> .
<TOVO-N>           ::= <TOVO> .                                                  11.4
*                      (THE MAN) TO SEE
<FORTOVO-N>        ::= <FORTOVO> .                                               11.5
*                      (THE MAN) FOR YOU TO SEE
<ADJINRN>          ::= <LAR> / <LONR> .                                          11.6
*                      (THE PEOPLE) RESPONSIBLE; (A LINE) TWO INCHES LONG
<APPOS>            ::= <NSTG> .                                                  11.7
*                      (MY FRIEND) JOHN
*
* 12.  LN LEFT ADJUNCTS OF N
*
<TPOS>             ::= <LTR> / <WHLN> / <LNSR> / <HOWQSTG> .                     12.2
```

```
<LTR>           ::= [<LT>] <*T>C / <*D> .                                    12.3
*                   ALL THE; ONLY
<LNSR>          ::= [<LN>] <*NS>C .                                          12.4
*                   MY YOUNG SON'S
<WHLN>          ::= WHOSE / WHICH / WHAT / <HOWQASTG> .                      12.5
<HOWQASTG>      ::= HOW <NULHOWQASTG>C [OF] <*T>C .                          12.6
<NULHOWQASTG>   ::= MUCH / MANY / <*ADJ> .                                   12.6A
<HOWQSTG>       ::= HOW <NULHOWQSTG>C .                                      12.7
<NULHOWQSTG>    ::= MUCH / MANY .                                            12.7A
<QPOS>          ::= <LQR> .                                                  12.8
<LQR>           ::= [<LQ>] <QVAR>C [<RQ>] .                                  12.9
<QVAR>          ::= <*Q> / <CPDNUMBR> .                                      12.10
<CPDNUMBR>      ::= <*Q> <*Q>C / <CPDNUMBR> <*Q>C .                          12.11
*                   TWO HUNDRED
<APOS>          ::= <ADJADJ> [<APOS>] .                                      12.12
<ADJADJ>        ::= <LAR1> / <LQNR> .                                        12.12A
<LAR1>          ::= [<LA>] <AVAR>C [<RA1>] .                                 12.13
<AVAR>          ::= [<LCDA>] <*ADJ>C / [<LCDVA>] <NULAVAR>C .                12.14
<NULAVAR>       ::= <*VING> / <*VEN> .                                       12.14A
*                   GREEN (TREE); GROWING (TREE); FALLEN (TREE)
<LQNR>          ::= [<LQN>] <QNPOS>C [<RQ>] .                                12.15
<QNPOS>         ::= <QN> / <NQ> .                                            12.16
<QN>            ::= <NUMBRSTG> <*N>C [<SCALESTG>] .                          12.17
*                   TWENTY-FIVE FOOT HIGH (TREE)
<NUMBRSTG>      ::= <*Q> / <CPDNUMBR> .                                      12.18
<SCALESTG>      ::= <*ADJ> / <PN> / OR SO .                                  12.19
<NQ>            ::= <*N> <NUMBRSTG> .                                        12.20
*                   SIZE TEN (DRESS)
<NPOS>          ::= [<*NS>] [<NPOS1>]C / = <NPOS1> <*NS>C .                  12.21
<NPOS1>         ::= [<LCDN>] <NULNPOS>C / <NPOS1> [<LCDN>] <NULNPOS>C .      12.22
<NULNPOS>       ::= <*N> / <*VING> .                                         12.22A
*                   FAMILY (TREE); LEARNING (TREE)
*
* 13.  RIGHT ADJUNCTS  -  OTHER THAN RN
*
<RW>            ::= <DSTG> .                                                 13.1
<RQ>            ::= OR SO / ENOUGH .                                         13.2
<RA>            ::= <SN> / <ASSERTION> / <PN> / <PVINGSTG> / <PSVINGO> /     13.3
                    <TOVC-N> / <FORTOVO-N> / <TOVO> / ENOUGH .
<RA1>           ::= ENOUGH .                                                 13.4
<RV>            ::= <REPRV> [<RV>] .                                         13.5
<REPRV>         ::= <DSTG> [<RD>] / <PN> / <QN> .                            13.5A
<RD>            ::= ENOUGH .                                                 13.6
*
* 14.  LEFT ADJUNCTS  --  OTHER THAN LN
*
<LW>            ::= <*D> .                                                   14.1
<LV>            ::= <DSTG> .                                                 14.2
<LQ>            ::= <DSTG> / [VERY] <NULLQ>C .                               14.3
<NULLQ>         ::= <*ADJ> / <*VING> / <*VEN> .                              14.3A
<LQN>           ::= <DSTG> .                                                 14.3B
<LA>            ::= <DSTG> .                                                 14.4
<LP>            ::= <DSTG> / <QNPOS> .                                       14.5
```

```
<LCS>           ::= <*D> .                                              14.6
<LPRC>          ::= <DSTG> .                                            14.7
<LT>            ::= <*D> / <*Q> .                                       14.8
<LCDA>          ::= <*N> .                                              14.9
<LCDVA>         ::= - <*N> / - <*D> / - <*ADJ> .                        14.10
<LCDN>          ::= <*ADJ> / - <*Q> .                                   14.11
*
* 15.  WH-STRINGS
*
<WHETHS>        ::= <NULWHETHS> [<ORNOT>] <ASSERTION>C [<ORNOT>] .      15.1
<NULWHETHS>     ::= WHETHER / WHERE / WHEN / HOW / WHY / IF .            15.1A
*                   WHETHER OR NOT I WILL VOTE (IS A GOOD QUESTION)
<WHETHTOVC>     ::= <NULWHETHS> <TOVC>C .                               15.2
*                   WHETHER OR NOT TO VOTE (IS A GOOD QUESTION)
<WHS-N>         ::= <NULWHS-N> <ASSERTION>C .                           15.3
<NULWHS-N>      ::= WHO / WHOM / WHICH / WHAT / WHOSE / THAN / AS .      15.3A
*                   WHOM WE SHOULD ELECT (IS A GOOD QUESTION)
<PWHS>          ::= <*P> <NULPWHS> <ASSERTION>C .                       15.4
<NULPWHS>       ::= WHOM / WHICH / WHAT / WHOSE / WHERE / WHEN .         15.4A
*                   WITH WHAT WE CAN BRIBE THE OFFICIAL (IS A GOOD QUESTION)
<PWHS-PN>       ::= <*P> <NULPWHS-PN> <ASSERTION>C .                    15.5
<NULPWHS-PN>    ::= WHOM / WHICH / WHAT / WHOSE .                       15.5A
*                   ON WHOM WE CAN DEPEND (IS A GOOD QUESTION)
<WHNS-N>        ::= <WHN> <ASSERTION>C .                                15.6
*                   WHICH CANDIDATE WE SHOULD ELECT (IS A GOOD QUESTION)
<PWHNS>         ::= <*P> <WHN> <ASSERTION>C .                           15.7
*                   WITH WHOSE MONEY WE CAN BRIBE THE OFFICIAL (IS A...)
<PWHNS-PN>      ::= <*P> <WHN> <ASSERTION>C .                           15.8
*                   ON WHICH CANDIDATE WE CAN DEPEND (IS A GOOD QUESTION)
<WHATS-N>       ::= WHAT <ASSERTION>C .                                 15.9
*                   WHAT WE PAID THE MAN (WAS TOO MUCH)
<WHERES>        ::= WHERE <ASSERTION>C .                                15.10
*                   WHERE WE MET HIM (WILL REMAIN A SECRET)
<WHENS>         ::= [<NULWHENS>] <ASSERTION>C .                         15.11
<NULWHENS>      ::= WHERE / WHEN .                                      15.11A
*                   (THE DAY) WHEN WE WON; (THE DAY) WE WON
<THATS-N>       ::= THAT <ASSERTION>C .                                 15.12
*                   (THE OFFICIAL) THAT WE BRIBED
<S-N>           ::= <ASSERTION> .                                       15.13
*                   (THE OFFICIAL) WE BRIBED
<SAWHICHSTG>    ::= <NULSAWHICHSTG> <ASSERTION>C .                      15.14
<NULSAWHICHSTG> ::= WHICH / THAN / AS .                                 15.14A
*                   (HE SQUEALED,) WHICH SURPRISED US
<WHEVERS-N>     ::= <NULWHEVERS-N> <ASSERTION>C .                       15.15
<NULWHEVERS-N>  ::= WHOEVER / WHOMEVER / WHICHEVER / WHATEVER /         15.15A
                    WHOSOEVER .
*                   WHATEVER THE JUDGE GIVES US (IS WELL DESERVED)
```

INDEX OF NON-TERMINAL SYMBOLS

<ADJADJ>	12.12A
<ADJINRN>	11.6
<ADJN>	8D.4
<ANNVAR>	5.3C
<APOS>	12.12
<APPOS>	11.7
<ASOBJBE>	8D.7
<ASSERTION>	2.1
<ASSERTIONVO>	2.1A
<ASTG>	8.8
<AVAR>	12.14
<BEINGO>	8.5
<CISHOULD>	6.5
<CENTER>	1.3
<CPDNUMBR>	12.11
<CSSTG>	9.7
<DP1>	8B.1
<DP1P>	8B.10
<DP1PN>	8B.6
<DP2>	8B.2
<DP2PN>	8B.7
<DP3>	8B.3
<DP3PN>	8B.8
<DP4>	8B.4
<DP4PN>	8B.9
<DPSN>	8B.5
<DSTG>	9.2
<ENDMARK>	1.4
<FORTOVO>	6.2
<FORTOVO-N>	11.5
<HOWQASTG>	12.6
<HOWQSTG>	12.7
<INTRODUCER>	1.2
<LA>	14.4
<LAR>	8.9
<LAR1>	12.13
<LCDA>	14.9
<LCDN>	14.11
<LCDVA>	14.10
<LCS>	14.6
<LN1>	5.3E
<LNNVAR>	5.3A
<LNR>	5.3
<LNSR>	12.4
<LP>	14.5
<LPRO>	14.7
<LPRON>	5.7
<LQ>	14.3
<LQNR>	12.15
<LQR>	12.9
<LT>	14.8
<LTR>	12.3
<LTVR>	7.2
<LV>	14.2
<LVENR>	7.6
<LVINGR>	7.10
<LVR>	7.4
<LVSA>	7.8
<LW>	14.1
<LWR>	7.3
<NA>	8D.3
<NAMESTG>	5.6
<ND>	8D.5
<NN>	8.7
<NNVAR>	5.3D
<NPN>	8A.8
<NPOS>	12.21
<NPOS1>	12.22
<NPSNWH>	8A.15
<NPSVINGO>	8A.13
<NPVINGSTG>	8A.12
<NPVINGO>	8A.14
<NQ>	12.20
<NSNWH>	8C.4
<NSTG>	5.2
<NSTGO>	8.6
<NSTGT>	9.4
<NSVINGO>	5.9
<NTHATS>	8C.3
<NTOVO>	8C.1
<NUL1WHQ-N>	3.3A
<NUL1RSUBJ>	9.5A
<NUL2WHQ-N>	3.3B
<NUL2RSUBJ>	9.5B
<NULAVAR>	12.14A
<NULBEINGO>	8.5A
<NULDPPN>	8B.6A
<NULHOWQASTG>	12.6A
<NULHOWQSTG>	12.7A
<NULLQ>	14.3A
<NULNAMESTG>	5.6A
<NULNV>	8B.4A
<NULNTHATS>	8C.3A
<NULNPOS>	12.22A

<NULOBES> 4,2A
<NULPWHQ> 3,6A
<NULPWHS> 15,4A
<NULPWHS-PN> 15,5A
<NULSASOBJBE> 8D,6A
<NULSAWHICHSTG> 15,14A
<NULSOBJBESA> 9,9A
<NULSUB2> 10,3A
<NULSUB8> 10,9A
<NULSVINGC> 6,3A
<NULTSUBJVO> 4,1A
<NULVINGSTGPN> 8A,10A
<NULWHENS> 15,11A
<NULWHETHS> 15,1A
<NULWHEVERS-N> 15,15A
<NULWHQ> 3,4A
<NULWHS-N> 15,3A
<NUMBRSTG> 12,18
<NVAR> 5,4
<NVSA> 9,11
<NWHSTG> 5,5
<OBES> 4,2
<OBJBE> 8,4
<OBJBESA> 9,8
<OBJECT> 8,1
<OBJECTBE> 8,3
<ORNOT> 3,2
<P1> 8A,1
<PA> 9,3
<PASSOBJ> 8,2
<PERMUTATION> 2,4
<PN> 8A,3
<PNN> 8A,9
<PNSNWH> 8C,7
<PNTHATS> 8C,5
<PNTHATSVC> 8C,6
<PNVINGSTG> 8A,11
<PROSENT> 2,3
<PSNWH> 8A,7
<PSTG> 8A,2
<PSVINGO> 8A,5
<PVINGO> 8A,6
<PVINGSTG> 8A,4
<PWHNQ> 3,9
<PWHNQ-PN> 3,8
<PWHNS> 15,7
<PWHNS-PN> 15,8
<PWHQ> 3,6
<PWHQ-PN> 3,5
<PWHS> 15,4
<PWHS-PN> 15,5
<QN> 12,17
<QNPOS> 12,16
<QPOS> 12,8
<QUESTION> 2,2
<QVAR> 12,10
<RA> 13,3
<RA1> 13,4
<RD> 13,6
<REPRN> 11,1A
<REPRV> 13,5A
<REPSA> 9,1A
<REPSARS> 9,1A
<RN> 11,1
<RNP> 11,2
<RNSUBJ> 9,6
<RNWH> 11,3
<RQ> 13,2
<RSUBJ> 9,5
<RV> 13,5
<RW> 13,1
<S-N> 15,13
<SA> 9,1
<SARS> 9,1
<SASOBJBE> 8D,6
<SAWH> 9,10
<SAWHICHSTG> 15,14
<SCALESTG> 12,19
<SENTENCE> 1,1
<SN> 5,11
<SNWH> 6,6
<SOBJBE> 8D,1
<SOBJBESA> 9,9
<STOVO-N> 8C,8
<SUB0> 10,1
<SUB1> 10,2
<SUB2> 10,3
<SUB3> 10,4
<SUB4> 10,5
<SUB5> 10,6
<SUB6> 10,7
<SUB7> 10,8
<SUB8> 10,9
<SUBJECT> 5,1
<SVEN> 8D,2
<SVINGO> 6,3
<SVO> 8C,2
<THATS> 6,1
<THATS-N> 15,12
<THATVING> 6,4
<TOVO> 7,12
<TOVO-N> 11,4
<TPOS> 12,2
<TQPOS> 5,3B
<TSUBJVO> 4,1
<TSUBJVOVO> 4,1B
<VENO> 7,5
<VENPASS> 7,7
<VERB> 7,1
<VERB1> 7,13

<VERB2>	7.14
<VINGC>	7.9
<VINGCFN>	5.10
<VINGSTG>	5.8
<VINGSTGPN>	8A.10
<VC>	7.11
<WHATS-N>	15.9
<WHENS>	15.11
<WHERES>	15.10
<WHETHS>	15.1
<WHETHTOVC>	15.2
<WHEVERS-N>	15.15
<WHLN>	12.5
<WHN>	3.9A
<WHNQ-N>	3.7
<WHNS-N>	15.6
<WHQ>	3.4
<WHQ-N>	3.3
<WHS-N>	15.3
<YESNOQ>	3.1
<YESNOQVO>	3.1A

```
*
*                        APPENDIX  B
*
*               THE RESTRICTIONS EXPRESSED IN THE
*                BOTTOM-UP RESTRICTION LANGUAGE
*
*
*      DEEPEST VERB AND DEEPEST OBJECT OF "BE" RESTRICTIONS
*
*  THESE RESTRICTIONS GOVERN WHETHER THE DEEPEST VERB OR DEEPEST OBJECT OF
*  "BE" IS BROUGHT UP AS THE CORE OF THE VERB PHRASE.
*
*  SET DOBFLG WHEN OBJBE OCCURS.
DVCB1:  HOUSED AT <OBJBE>:  <PASSOBJ>; <OBJECT>; <ASOBJBE>; <OBJECTBE>:
                  CORE HAS DOBFLG.
                  CORE HAS ASTGOBFLG, SINGFLG, PLURALFLG OF CORE2.
        HOUSED AT <ASOBJBE>:  <OBJECT>; <PASSOBJ>;
                  <OBJECTBE>:  <OBJECT>:
                    CORE HAS DOBFLG, ASTGOBFLG, SINGFLG, PLURALFLG OF
                         CORE2.
*  SET DVBFLG WHEN A VERBAL OCCURS.
DVCB2:  HOUSED AT <LTVR>:  <VERB1>;
                  <VERB>:  <ASSERTIONVO>;
                  <LVINGR>:  <VINGO>;
                  <*VING>:  <VINGOFN>;
                  <LVENR>:  <VENO>; <VENPASS>;
                  <LVR>:  <VO>; <TOVO>;
                  <VERB2>:  <TSUBJVOVO>; <YESNOQVO>;
                  <VERB3>:  <OBES>:
                    CORE HAS DVBFLG.
        HOUSED AT <VERB1>:  <YESNOQ>; <TSUBJVO>:
                    CORE HAS DVBFLG OF CORE2.
*  SET VBOBJFLG WHEN OBJECT OR PASSOBJ HAS A V-O STG AS ITS VALUE.
DVCB3:  HOUSED AT <TOVO>, <VENO>, <VINGO>, <VO>:  <OBJECT>;
                  <VENPASS>, <VINGO>, <TOVO>:  <OBJECTBE>;
                  <VINGO>, <TOVO>, <VENPASS>, <VO>:  <PASSOBJ>:
                    CORE HAS VBOBJFLG.
        HOUSED AT <OBJECTBE>:  <OBJECT>:
                    CORE HAS VBOBJFLG OF CORE2.
*  IF OBJECT IS OBJBE PASS UP DOBFLG AND CORE2 BECOMES NEW CORE.  ANY
*  OTHER VALUE EXCEPT V-O STGS, PASS UP DVBFLG AND TAKE CORE1 AS NEW CORE.
*  FOR V-O STGS TAKE CORE2 AND WHICHEVER FLAG IT HAS.
DVCB4:  HOUSED AT <OBJECT>:  <VO>; <VINGO>; <VENO>; <TOVO>; <VINGOFN>;
                             <ASSERTIONVO>; <YESNOQVO>; <TSUBJVOVO>;
                  <PASSOBJ>:  <VENPASS>:
                    IF CORE2 HAS NOT VBOBJFLG AND DOBFLG AND CORE1 HAS
                         NOT V OR VBE OR BEREP THEN CORE HAS ATTRIBUTES
                         OF CORE2 AND CORE HAS DOBFLG, ASTGOBFLG,
                         SINGLFG, PLURALFLG OF CORE2.
                    IF CORE2 HAS NOT VBOBJFLG AND (CORE2 HAS NOT DOBFLG
                         OR CORE1 HAS V AND NOT (VBE OR BEREP) THEN
                         CORE HAS ATTRIBUTES OF CORE1 AND CORE HAS
                         DVBFLG, DVBVENPASSFLG OF CORE1.
```

```
                    IF CORE2 HAS VBOBJFLG THEN CORE HAS ATTRIBUTES OF
                         CORE2 AND CORE HAS DVBFLG, DVBVENPASSFLG,
                         DOBFLG, ASTGOBFLG, SINGFLG, PLURALFLG OF
                         CORE2.
*  PASS DVBFLG OR DOBFLG UP TO THE OBJECT OR PASSOBJ.
DVOBS:  HOUSED AT <VO>, <VINGO>, <VENO>, <TOVO>, <OBJECTBE>:  <OBJECT>;
                  <VENPASS>, <VO>, <VINGO>, <TOVO>:  <PASSOBJ>;
                  <VENPASS>, <VINGO>, <TOVO>:  <OBJECTBE>;
                  <VENPASS>:  <NULOBES>;
                  <VINGO>:  <NULSASOBJBE>;
                  <TOVC>:  <TOVO-N>;
                  <ASSERTIONVO>:  <ASSERTION>:
                    CORE HAS DVBFLG, DOBFLG, ASTGOBFLG, SINGFLG,
                         PLURALFLG OF CORE2.
```

```
*
*                  AGREEMENT RESTRICTIONS
*
* WAGREE1:  SUBJECT AND VERB AGREE IN NUMBER.
WAGREE1:  HOUSED AT <OBJECT>:  <ASSERTIONVO>;
                   <LVR>: <VERB>,2,3;
                     IF CORE1 HAS SINGULAR OR SINGFLG THEN CORE HAS
                          SINGFLG.
                     IF CORE1 HAS PLURAL OR PLURALFLG THEN CORE HAS
                          PLURALFLG.
          HOUSED AT <NAMESTG>, <NWHSTG>:  <NSTG>;
                   <VINGSTG>, <SN>:  <SUBJECT>:
                     CORE HAS SINGFLG.
          HOUSED AT <OPDNUMBR>:  <QVAR>:
                     CORE HAS PLURALFLG.
          HOUSED AT <QVAR>:  <LQR>;
                   <LQR>:  <QPOS>;
                     CORE HAS PLURALFLG OF CORE2.
          HOUSED AT <QPOS>:  <LN1>; <TQPOS>; <LN2>;
                   <TQPOS>:  <LNNVAR>:
                     IF CORE2 HAS PLURAL OR PLURLAFLG THEN CORE HAS
                          PLURALFLG.
                     IF CORE2 HAS SINGULAR THEN CORE HAS SINGFLG.
          HOUSED AT <LN1>, <LNNVAR>:  <LNR>;
                   <LNR>:  <NSTG>;
                   <NSTG>:  <SUBJECT>;
                   <SUBJECT>:  <ASSERTION>;
                   <VERB>:  <ASSERTIONVO>:
                     CORE HAS PLURALFLG, SINGFLG OF CORE2.
          HOUSED AT <ASSERTIONVO>:  <ASSERTION>:
             $SS  OR  IF CORE2 HAS SINGULAR OR SINGFLG THEN CORE1 HASNT
                          PLURAL OR PLURALFLG.
             $SS  OR  IF CORE2 HAS PLURAL OR PLURALFLG THEN CORE1 HAS
                          NOT (SINGULAR OR SINGFLG) OR AGGREGATE OR
                          !I!.
* WAGREE2:  SUBJECT AND NOUN OBJECT OF !BE! AGREE IN NUMBER.
WAGREE2:  HOUSED AT <NSTG>:  <OBJBE>:
                     CORE HAS SINGFLG, PLURALFLG OF CORE2.
          HOUSED AT <ASSERTIONVO>:  <ASSERTION>;
                   <YESNOQVO>:  <YESNOQ>;
                   <TSUBJVOVO>:  <TSUBJVO>;
                   <VINGO>:  <SVINGO>;
                   <TOVO>:  <FORTOVO>; <NTOVO>;
                   <OBJBE>:  <SOBJBE>;
                   <NULSASOBJBE>:  <SASOBJBE>;
                   <TOVO-N>:  <STOVO-N>:
             $RARE OR IF CORE1 HASNT !IT! AND CORE2 HAS DOBFLG THEN
                          (IF CORE2 HAS (SINGULAR OR SINGFLG) AND
                          NOT AGGREGATE THEN CORE1 HASNT PLURAL
                          OR PLURALFLG) AND (IF CORE2 HAS PLURAL OR
                          PLURALFLG THEN CORE1 HAS NOT (SINGULAR OR
                          SINGFLG) OR AGGREGATE).
          HOUSED AT <SUBJECT>:  <OBES>:
```

```
            $RARE OR IF CORE2 HASNT 'IT' AND CORE1 HAS DOBFLG THEN
                        (IF CORE1 HAS (SINGULAR OR SINGFLG) AND
                        NOT AGGREGATE THEN CORE2 HASNT PLURAL
                        OR PLURALFLG) AND (IF CORE1 HAS PLURAL OR
                        PLURALFLG THEN CORE2 HAS NOT (SINGULAR OR
                        SINGFLG) OR AGGREGATE).
* A RECIPROCAL OR COLLECTIVE VERB OCCURING WITH A NULL OBJECT CANNOT
* HAVE A SINGULAR SUBJECT.
WAGREE3:  HOUSED AT <ASSERTIONVO>:  <ASSERTION>;
                    <YESNOQVC>:  <YESNOQ>;
                    <TSUBJVOVO>:  <TSUBJVO>;
                    <VINGC>:  <SVINGC>;
                    <TOVO>:  <FORTOVC>; <NTOVO>;
                    <OBJBE>:  <SOBJBE>;
                    <NULSASOBJBE>:  <SASOBJBE>;
                    <TOVO-N>:  <STOVO-N>:
            $WS  OR  IF CORE2 HAS NULLOBJFLG AND VRECIP THEN CORE1 HAS
                        NOT (SINGULAR OR SINGFLG) OR AGGREGATE.
            $WS  OR  IF CORE2 HAS NULLOBJFLG AND VCOLLECTIVE THEN
                        CORE1 HAS NOT (SINGULAR OR SINGFLG) OR
                        AGGREGATE OR COLLECTIVE.
* WAGREE4:  LN AND N AGREE IN NUMBER .
WAGREE4:  HOUSED AT <QPOS>:  <TQPOS>; <LN1>; <LN2>;
                    <ANNVAR>:  <LNVAR>:
            $SS  OR  IF CORE2 HAS PLURAL OR PLURALFLG THEN CORE1 HASNT
                        SINGULAR OR SINGFLG.
            $SS  OR  IF CORE2 HAS SINGULAR OR SINGFLG THEN CORE1 HASNT
                        PLURAL OR PLURALFLG.
* WAGREE5:  HOST AND APPOSITION NOUN AGREE IN NUMBER .
WAGREE5:  HOUSED AT <APPOS>:  <REPRN>:
                    CORE HAS APPOSFLG.
          HOUSED AT <REPRN>:  <RN>:
                    CORE HAS APPOSFLG OF CORE2.
          HOUSED AT <RN>:  <LNR>,1,2; <LNR>,2,2; <LPRON>:
             $SS  OR  IF CORE2 HAS APPOSFLG AND (SINGULAR OR SINGFLG)
                         THEN CORE1 HASNT PLURAL OR PLURALFLG.
             $SS  OR  IF CORE2 HAS APPOSFLG AND (PLURAL OR PLURALFLG)
                        THEN CORE1 HAS NOT (SINGULAR OR SINGFLG)
                        OR AGGREGATE.
* WAGREE6:  IN SAWHICHSTG, IF THE SUBJECT HAS BEEN OMITTED, THE TENSED VERB
* IS NOT PLURAL (THEY  ARE NOT HERE, WHICH IS SURPRISING).
WAGREE6:  HOUSED AT <ASSERTIONVO>:  <ASSERTION>:
                    CORE HAS PLURALFLG OF CORE2.
          HOUSED AT <ASSERTION>:  <SAWHICHSTG>:
             $SS  OR  IF CORE2 HAS NULLSUBJFLG THEN CORE2 HASNT PLURAL
                        OR PLURALFLG.
* WAGREE7:  AGREEMENT OF ROVING QUANTIFIERS AND REFLEXIVES WITH THE
*           NOUN THEY ADJOIN (THEY ARE ALL IN AGREEMENT, WE WERE NOT
*           THERE OURSELVES)
WAGREE7:  HOUSED AT <RSUBJ>:  <REPSA>:
                    IF CORE2 HAS SINGULAR OR SINGFLG THEN CORE HAS
                        RSSINGFLG.
```

```
                    IF CORE2 HAS PLURAL OR PLURALFLG THEN CORE HAS
                      RSPLURFLG.
          HOUSED AT <REPSA>:  <SA>;
                    <SA>: <ASSERTIONVO>,5; <YESNOQVO>,4; <TSUBJVOVO>,4;
                          <TOVO>,7; <VINGO>,6; <NULSASOBJBE>;
                          <VENPASS>,6;
                    <TOVO>:  <TOVO-N>;
                    <VINGO>:  <NULSASOBJBE>:
                      CORE HAS RSSINGFLG, RSPLURFLG OF CORE2.
          HOUSED AT <SA>:  <SOBJBE>,4;
                    <ASSERTIONVO>:  <ASSERTION>;
                    <YESNOQVOVO>:  <YESNOQ>;
                    <TOVO>:  <NTOVO>;
                    <TOVO-N>:  <STOVO-N>;
                    <NULSASOBJBE>:  <SASOBJBE>;
                    <VENPASS>:  <SVEN>:
             $SS  OR  IF CORE2 HAS RSSINGFLG THEN CORE1 HASNT PLURAL
                      OR PLURALFLG,
             $SS  OR  IF CORE2 HAS RSPLURFLG THEN CORE1 HAS NOT
                      (SINGULAR OR SINGFLG) OR AGGREGATE.
* WAGREE8: IF THE PREPOSITION IN PN, PWHN-PN OR PWHN IS !BETWEEN! OR !AMONG!
* THEN THE NOUN MUST BE PLURAL (BETWEEN THE TREES *AMONG THE TREE).
WAGREE8:  HOUSED AT <VINGOFN>:  <WHN>;
                    <NWHSTG>, <NAMESTG>:  <NSTG>:
                      CORE HAS STGFLG.
          HOUSED AT <NSTG>:  <NSTGO>:
                      CORE HAS STGFLG OF CORE2.
          HOUSED AT <NSTGO>:  <PN>;
                    <WHN>:  <PWHNS-PN>; <PWHNS>; <PWHNQ-PN>; <PWHNQ>:
             $SS  OR  IF CORE1 HAS !BETWEEN! THEN CORE2 HAS (NOT
                      (SINGULAR OR SINGFLG) OR AGGREGATE)
                      AND NOT STGFLG.
             $SS  OR  IF CORE1 HAS !AMONG! THEN CORE2 HAS (NOT
                      (SINGULAR OR SINGFLG) OR AGGREGATE)
                      AND NOT STGFLG AND NOT !TWO!.
```

```
*
*                    COMMA RESTRICTIONS
*
* DCCM1:  THE LISTED STRINGS ARE NOT TAKEN AS INSTANCES OF RN WHEN PRECEDED
* BY COMMA. (WE FOUND THE RIGHT FOOD TO EAT (RN), CONTRAST: WE FOUND THE
* RIGHT FOOD, TO EAT (SA)).
DCCM1:  HOUSED AT <S-N>, <TOVC>, <TOVO-N>, !WHATSOEVER!:  <REPRN>:
           $PUNC OR INTER(P) % !,!.
* DCCM2:  CONDITIONS FOR COMMA PRECEDING VERB
DCCM2:  HOUSED AT <RN>:  <LNR>,1,2; <LNR>,2,2;
                  <SA>:  <ASSERTION>,3; <NVSA>:
                    CORE HAS INTADJFLG.
        HOUSED AT <LNR>:  <NSTG>;
                  <NSTG>:  <SUBJECT>;
                  <SUBJECT>:  <ASSERTION>:
                    CORE HAS INTADJFLG OF CORE2.
        HOUSED AT <ASSERTIONVO>:  <ASSERTION>;
                  <VERB>:  <NVSA>:
           $PUNC OR IF INTER(P) = !,! THEN CORE1 HAS INTADJFLG.
* DCCM3: COMMA PRECEDES CERTAIN OPTIONS OF SA UNLESS THEY ARE SENTENCE
* INITIAL.   COMMA FOLLOWS INITIAL OCCURRENCES.
DCCM3:  HOUSED AT <OBJBESA>, <SOBJBESA>, <VINGO>, <VENPASS>: <REPSA>:
           $PUNC OR IF P % 0 THEN INTER(P) = !,!.
           $PUNC OR IF P = 0 THEN INTER(Q) = !,!.
* DCCM4: IN RN APPOSITION STRING ADJOINED TO NOUN PHRASE RARE UNLESS
* PRECEDED BY COMMA. ("HANK, THE MAN WE MET YESTERDAY IS A PAINTER", MORE
* RARE, "HANK THE MAN WE MET YESTERDAY IS A PAINTER").
DCCM4:  HOUSED AT <RN>:  <LNR>,1,2; <LNR>,2,2:
           $RARE OR IF CORE2 HAS APPOSFLG AND INTER(P) = !,! THEN
                       CORE1 HAS PRO.
* WCCM2:   NO COMMA PRECEDING N OR VING UNLESS THE LEFT ADJUNCT IS EMPTY
WCCM2:  HOUSED AT <NVAR>:  <NNVAR>:
           $PUNC OR IF INTER(P) = !,! THEN CORE1 HASNT NPOSFLG.
        HOUSED AT <NNVAR>:  <ANNVAR>:
           $PUNC OR IF INTER(P) = !,! AND CORE2 HASNT NPOSFLG THEN
                       CORE1 HASNT APOSFLG.
                     CORE HAS NPOSFLG OF CORE2.
        HOUSED AT <ANNVAR>:  <LNVAR>:
           $PUNC OR IF INTER(P) = !,! AND CORE2 HASNT APOSFLG OR
                       NPOSFLG THEN CORE1 HASNT TPOSFLG OR QPOSFLG.
* WCCM3:  IF PUNCTUATION COMMA PRECEDES THESE CERTAIN OPTIONS OF RN THEN
* COMMA OR ENDMARK FOLLOWS.
WCCM3:  HOUSED AT <VENPASS>, <VINGO>, <ADJINRN>, <APPOS>:  <REPRN>:
           $PUNC OR IF INTER(P) = !,! THEN INTER(Q) = !,! OR WORD(Q+1)
                       = !.! OR !;! OR !?!.
* WCCM4:  CONDITIONS FOR COMMA PRECEDING OBJECT.
WCCM4:  HOUSED AT <SA>:  <YESNOQ>,5; <TSUBJVO>,5; <ASSERTIONVO>,2;
                         <VENPASS>,3; <VINGO>,3; <TOVO>,4; <VENO>,2;
```

```
                        <VO>,2; <VERB2>:
                    CORE HAS INTSAFLG.
                    CORE HAS SHORTSAFLG OF CORE2.
        HOUSED AT <RV>:  <LVENR>; <LTVR>; <LVINGR>; <LVR>:
                    CORE HAS INTRVFLG.
        HOUSED AT <LVENR>:  <VENO>; <VENPASS>;
                  <LVINGR>:  <VINGO>;
                  <LVR>:  <TOVO>; <VO>; <VERB2>; <VERB>,2,3; <VERB>,3,1;
                  <LTVR>:  <VERB>;
                  <VERB>:  <ASSERTIONVO>:
                    CORE HAS INTRVFLG OF CORE2.
        HOUSED AT <VERB2>:  <YESNOQVO>; <TSUBJVOVO>:
                    CORE HAS INTSAFLG, INTRVFLG, SHORTSAFLG OF CORE2.
        HOUSED AT <YESNOQVO>:  <YESNOQ>;
                  <TSUBJVOVO>:  <TSUBJVO>;
                  <OBJECT>:  <YESNOQVO>; <TSUBJVOVO>; <ASSERTIONVO>;
                             <VENO>; <VINGO>; <VO>; <TOVO>;
                  <PASSOBJ>:  <VENPASS>:
           $PUNC OR IF INTER(P) = !,! THEN CORE1 HAS INTSAFLG OR
                             INTRVFLG.
* WCCM5: THERE IS NOT A LONG SA BEFORE AN OBJECT UNLESS A COMMA PRECEDES.
WCCM5:  HOUSED AT <DSTG>, <*INT>, <PN>, <PA>, <NSTGT>:  <REPSA>:
                    CORE HAS SHORTSAFLG.
        HOUSED AT <REPSA>:  <SA>:
                    CORE HAS SHORTSAFLG OF CORE2.
        HOUSED AT <OBJECT>:  <YESNOQVO>; <TSUBJVOVO>; <ASSERTIONVO>;
                             <VENO>; <VINGO>; <VO>; <TOVO>:
           $PUNC OR IF CORE1 HAS INTSAFLG THEN CORE1 HAS SHORTSAFLG OR
                             INTER(P) = !,!.
* WCCM7: NON-ZERO VALUE OF LP IS RARE, UNLESS PRECEDED BY A COMMA, IN
* PRE-OBJECT PN SENTENCE ADJUNCT ("HE DOESN'T READ, EXACTLY LIKE HIS FATHER,
* MANY BOOKS", *"HE DOESN'T EXACLTLY LIKE THE FRENCH PLAY" WITH "EXACTLY"=LP
WCCM7:  HOUSED AT <LP>:  <PN>:
                    CORE HAS LPFLG.
        HOUSED AT <PN>: <REPSA>;
                  <REPSA>:  <SA>:
                    CORE HAS LPFLG OF CORE2.
        HOUSED AT <RW>:  <YESNOQ>:
                    CORE HAS INTRVFLG.
        HOUSED AT <SA>:  <YESNOQ>,5; <ASSERTIONVO>,2; <VERB2>:
                    IF CORE2 HAS LPFLG THEN INTER(P) = !,! OR CORE1
                             HAS INTRVFLG.
* WCCM8:  IN THE STRING WHENS IN RN, IF THE STRING HEAD IS NULL, THE
*       PREVIOUS WORD CANNOT BE COMMA, *"WE KNOW THE DAY, HE ARRIVED."
*       AND IF THE STRING HEAD IS !WHEN! THE HOST NOUN MUST BE A TIME
*       NOUN.
WCCM8:  HOUSED AT !WHEN!:  <NULWHENS>:
           $PUNC OR INTER(P) % !,!.
                    CORE HAS WHENFLG.
        HOUSED AT <ASSERTION>:  <WHENS>:
           $PUNC OR INTER(P) % !,!.
        HOUSED AT <NULWHENS>:  <WHENS>;
```

```
          <WHENS>:  <RNWH>;
          <RNWH>:  <REPRN>;
          <REPRN>:  <RN>:
            CORE HAS WHENFLG OF CORE2.
HOUSED AT <RN>:  <LNR>,1,2:
   $POS4 OR IF CORE2 HAS WHENFLG THEN CORE1 HAS NTIME1.
HOUSED AT <RN>:  <LNR>,2,2; <LPROR>:
   $POS4 OR CORE2 HASNT WHENFLG.
```

```
*
*                    NOUN PHRASE RESTRICTIONS
*
* DN1:  AMONG THE LEFT QUANTIFIERS OF NOUN, CERTAIN ARTICLES, QUANTIFIERS OR
* ADJECTIVES MAY BE USED WITHOUT A FOLLOWING NOUN IN THE STRING POSITIONS
* OCCUPIED BY NOUN ("THESE WERE NOT USED", "FIVE WERE REJECTED").
DN1:   HOUSED AT <APOS>:  <LN1>:
          $PCS2 OR CORE1 HAS TPOSFLG OR QPOSFLG,
          $PCS4 OR (CORE2 HAS NQFLG AND CORE1 HAS TPOSFLG OR
                        QPOSFLG) OR (CORE2 HAS VEN OR VING OR APREQ
                        OR COMPARATIVE OR SUPERLATIVE AND CORE1 HAS NS
                        OR TDEF OR TPOSS),
       HOUSED AT <NQ>:  <QNPCS>:
                  CORE HAS NQFLG,
       HOUSED AT <QNPCS>:  <LQNR>;
                 <LQNR>:  <ADJADJ>;
                 <ADJADJ>:  <APOS>:
                  CORE HAS NQFLG OF CORE2,
       HOUSED AT <TPOS>:  <LN1>; <TQPOS>:
                  CORE HAS TPOSFLG,
       HOUSED AT <QPOS>:  <LN1>; <TQPOS>:
                  CORE HAS QPOSFLG,
       HOUSED AT <APOS>:  <LN1>; <ANNVAR>:
                  CORE HAS APOSFLG,
       HOUSED AT <HOWQSTG>:  <TPOS>;
                 <HOWQASTG>:  <WHLN>:
                  CORE HAS HOWQFLG,
       HOUSED AT <WHLN>:  <TPOS>;
                 <TPOS>:  <LN1>:
                  CORE HAS HOWQFLG OF CORE2,
       HOUSED AT <LN1>:  <LNR>:
          $PCS4 OR CORE2 HAS APOSFLG OR (QPOSFLG AND NOT APOSFLG)
                        OR (NOT QPOSFLG AND NOT APOSFLG AND (NS OR
                        TDEM OR TQUAN OR HOWQFLG)),
* DN3:  A POSSESSIVE PROPER NAME NS:NAME DOES NOT QUALIFY IN NPOS ("A
* CHILDREN'S CHOIR", *"A JOHN'S CHOIR").
DN3:   HOUSED AT <*NS>:  <NPCS>,1,1; <NPOS>,2,2:
          $PCS4 OR CORE2 HASNT NAME,
* DN50:  S-N DISQUALIFIED AS VALUE OF RN IN NOUN PHRASE WHOSE CORE VALUE
* IS ZERO, UNLESS LEFTMOST ELEMENT OF LEFT ADJUNCT IS (1) A QUANTIFIER, (2)
* ARTICLE IN SUBCLASS TDEM (THESE, THOSE), OR (3) RARELY, AN ARTICLE IN
* TQUAN (ANY, SUCH, SOME, EITHER, NEITHER) ("MANY YOU SAW ARE NEW",
* *"JOHN'S HE FIXED WAS A MODEL T").
DN50:  HOUSED AT <S-N>:  <REPRN>:
                  CORE HAS S-NFLG,
       HOUSED AT <REPRN>:  <RN>:
                  CORE HAS S-NFLG OF CORE2,
       HOUSED AT <LN1>:  <LNR>,2,1:
                  CORE HAS TPOSFLG, QPOSFLG, APOSFLG OF CORE2,
```

```
       HOUSED AT <RN>:  <LNR>,2,2:
          $PCS2 OR IF CORE2 HAS S-NFLG THEN CORE1 HASNT APOSFLG,
          $RARE OR IF CORE2 HAS S-NFLG THEN CORE1 HAS QPOSFLG OR (TDEM
                       AND PLURAL),
          $PCS4 OR IF CORE2 HAS S-NFLG THEN CORE1 HAS QPOSFLG OR (TDEM
                       AND PLURAL) OR TQUAN.
*  DN51:  IF S-N ADJOINS A NOUN PHRASE, THEN IN THE LEFT ADJUNCT OF THE HOST
*         NOUN EITHER THE QUANTIFIER POSITION IS NOT EMPTY OR THERE IS AN
*         ARTICLE.
DN51:  HOUSED AT <TQPOS>:  <LNVAR>;
                 <LNVAR>:  <LNR>:
                   CORE HAS TPOSFLG, QPOSFLG OF CORE2.
       HOUSED AT <RN>:  <LNR>,1,2:
          $RARE OR IF CORE2 HAS S-NFLG THEN CORE1 HAS TPOSFLG OR
                       QPOSFLG.
* DN52:  IF THE APPOSITION STRING ADJOINS A PRONOUN, THE PRONOUN IS
*        EITHER "WE", "YOU", I.E. PRONOUNS WHICH ARE PLURAL AND
*        DEFINITELY HUMAN ("WE THE PEOPLE", *"HE THE INSTIGATOR") OR ELSE
*        THE PRONOUN IS AN OBJECT OF BE ("IT WAS HE THE INSTIGATOR OF ALL
*        OUR TROUBLES").
DN52:  HOUSED AT <RN>:  <LPRCR>:
                   IF CORE2 HAS APPOSFLG AND CORE1 HASNT NHUMAN AND
                       PLURAL THEN CORE HAS ONLYOBFLG.
       HOUSED AT <LPRCR>:  <NSTG>:
                   CORE HAS ONLYOBFLG OF CORE2.
       HOUSED AT <LPRCR>:  ALL OCCURRENCES EXCEPT <NSTG>;
                 <NSTG>:  ALL OCCURRENCES EXCEPT <OBJBE>:
          $PCS4 OR CORE2 HASNT ONLYOBFLG.
* DN53:  NO APPOSITION ON APPOSITION.
DN53:  HOUSED AT <RN>:  <LPRCR>; <LNR>,1,2; <LNR>,2,2;
                 <LNR>:  <NSTG>:
                   CORE HAS APPOSFLG OF CORE2.
       HOUSED AT <NSTG>:  <APPOS>:
          $PCS3 OR CORE2 HASNT APPOSFLG.
* DN54:  PN, DSTG, THATS-N CAN ADJOIN A PRONOUN IF IT IS !WE!, !YOU!, AN
*        INDEFINITE PRONOUN OR A PRONOUN WHICH IS AN OBJECT OF BE ("WE
*        HERE", "YOU OF THE COMMITTE", *"HE OF THE COMMITTEE", *"HE THAT
*        I LIKE").
DN54:  HOUSED AT <DSTG>, <THATS-N>, <PN>:  <REPRN>:
                   CORE HAS DTPFLG.
       HOUSED AT <REPRN>:  <RN>:
                   CORE HAS DTPFLG OF CORE2.
       HOUSED AT <RN>:  <LPRCR>:
                   IF CORE2 HAS DTPFLG AND CORE1 HASNT (NHUMAN AND
                       PLURAL) OR INDEFINITE THEN CORE HAS ONLYOBFLG.
* DN56:  IN RN, THE LISTED STRINGS DO NOT ADJOIN A NON-INDEFINITE PRONOUN
*        OR A PROPER NAME. (*"IT HE SAW FRIGHTENED HIM", "ANYONE SET UPON
*        REALIZING HIS OBJECTIVE CAN SUCCEED", *"HE SET UPON REALIZING
*        HIS OBJECTIVE SUCCEEDED").
DN56:  HOUSED AT <SN>, <ADJINRN>, <S-N>, <TOVO-N>, <TOVO>, <VENPASS>,
                 <VINGO>, "WHATSOEVER":  <REPRN>;
                 <PSNWH>, <PVINGSTG>, <PSVINGO>:  <RNP>:
                   CORE HAS INDNAMFLG.
```

```
        HOUSED AT <RNP>:  <REPRN>;
                  <REPRN>:  <RN>:
                    CORE HAS INDNAMFLG OF CORE2.
        HOUSED AT <RN>:  <LPRCR>:
        HOUSED AT <RN>:  <LNR>,1,2; <LNR>,2,2:
           $PCS4 OR IF CORE2 HAS INDNAMFLG THEN CORE1 HASNT N:NAME.
* WN1:  PERMISSIBLE QUANTIFIER-ARTICLE COMBINATIONS IN LTR ("SUCH A",
*        *"SUCH THE").
WN1:    HOUSED AT <LT>:  <LTR>:
           $PCS4 OR IF CORE2 HAS Q THEN CORE2 HAS QHALF OR QALL OR
                        QMANY.
        HOUSED AT <*T>:  <LTR>:
           $PCS4 OR IF CORE1 HAS QMANY THEN CORE2 HAS "A" OR "AN".
           $PCS4 OR IF CORE1 HAS QALL THEN CORE2 HASNT "A" OR "AN".
           $PCS4 OR IF CORE1 HAS Q THEN CORE2 HASNT "NO" OR TQUAN.
* WN3:  PRE-ARTICLE ADVERBS DLTPRO CAN OCCUR WITHOUT SUCCEEDING
*        ARTICLE (ONLY HE, EVEN MEN).
WN3:    HOUSED AT <*D>:  <LTR>:
           $PCS4 OR CORE2 HAS DLTPRO.
* WN4:  DLTPRO (ONLY A MAN, EVEN CHILDREN) IS NOT TAKEN AS LT IF THERE
*        IS NO T AND IT CAN BE TAKEN AS LQ ("ONLY TWO MEN ARRIVED").
WN4:    HOUSED AT <QPOS>:  <TQPOS>; <LN1>; <LN2>:
           $PCS4 OR CORE1 HASNT DLTPRO.
* WN5:  PRE-QUANTIFIER ADJ, VING OR VEN IS APREQ OR SUPERLATIVE ("AN
*        ADDITIONAL THREE PEOPLE", "THE TALLEST THREE BOYS").
WN5:    HOUSED AT <*ADJ>, <*VING>, <*VEN>:  <NULLQ>:
           $PCS4 OR CORE2 HAS APREQ OR SUPERLATIVE.
* WN6:  NOUN LEFT ADJUNCT OF NOUN IS NOT PLURAL OR ADJNOUN (*"A HOUSES
*        BOAT", "A FUNDAMENTAL THEOREM").
WN6:    HOUSED AT <*N>:  <LCDA>; <LCDVA>; <NULNPOS>:
           $PCS4 OR CORE2 HASNT PLURAL OR ADJNOUN.
* WN9:  A COUNT NOUN REQUIRES A PRECEDING ARTICLE, WH-WORD, QUANTIFIER
*        OR NS.  BUT IF THE COUNT NOUN IS ALSO SUBCLASSIFIED NCOUNT2,
*        N IN PN FOR APPROPRIATE P IS EXEMPTED; IF NCOUNT3, N IN OBJBE
*        IS EXEMPTED.
WN9:    HOUSED AT <QPOS>:  <TQPOS>:
                  IF CORE2 HAS Q AND SINGULAR AND QNUMBER THEN CORE
                       HAS CKCOUNTFLG.
        HOUSED AT <WHLN>:  <TPOS>:
                  CORE HAS CKCOUNTFLG.
        HOUSED AT <TPOS>:  <TQPOS>:
                  CORE HAS CKCOUNTFLG OF CORE2.
        HOUSED AT <ANNVAR>:  <LNVAR>:
           $WS  OR  IF CORE2 HAS SINGULAR AND NCOUNT1 AND NOT NCOUNT2
                       AND NOT NCOUNT3 THEN CORE1 HAS T OR NS OR
                       OKCOUNTFLG,
                    IF CORE2 HAS SINGULAR AND NCOUNT1 AND NCOUNT2 AND
                       CORE1 HASNT T OR NS OR OKCOUNTFLG THEN CORE
                       HAS ONLYPNFLG,
                    IF CORE2 HAS SINGULAR AND NCOUNT1 AND NCOUNT3 AND
                       CORE1 HASNT T OR NS OR OKCOUNTFLG THEN CORE
                       HAS ONLYOBFLG.
        HOUSED AT <LNNVAR>:  <LNR>;
```

```
                <LNR>:  <NSTG>:
                  CORE HAS ONLYOBFLG, ONLYPNFLG OF CORE2.
        HOUSED AT <LNR>:  ALL OCCURRENCES EXCEPT <NSTG>:
                <NSTG>:  ALL OCCURRENCES EXCEPT <OBJBE>; <NSTGO>:
           $WS  OR  CORE2 HASNT ONLYOBFLG OR ONLYPNFLG.
        HOUSED AT <NSTG>:  <OBJBE>:
           $WS  OR  IF CORE2 HAS ONLYPNFLG THEN CORE2 HAS ONLYOBFLG.
        HOUSED AT <NSTG>:  <NSTGO>:
           $WS  OR  IF CORE2 HAS ONLYOBFLG THEN CORE2 HAS ONLYPNFLG.
        HOUSED AT <NSTG>:  <NSTGO>:
                  CORE HAS ONLYPNFLG OF CORE2.
        HOUSED AT <NSTGO>:  ALL OCCURRENCES EXCEPT <PN>:
           $WS  OR  CORE2 HASNT ONLYOBFLG.
        HOUSED AT <NSTGO>:  <PN>:
           $PCS5 OR IF CORE2 HAS ONLYPNFLG THEN CORE1 EL NCOUNT2
                        (CORE2).
* WN10:  CONDITIONS FOR NOUN CORE = VING.
WN10:  HOUSED AT <NPOS>:  <NNVAR>:
                   CORE HAS NPOSFLG.
        HOUSED AT <NVAR>:  <NNVAR>:
           $PCS4 OR IF CORE2 HAS VING AND CORE1 HASNT NPOSFLG THEN
                        NULLOBJ EL OBJLIST(CORE2),
                     IF CORE2 HAS VING AND CORE1 HASNT NPOSFLG THEN
                        CORE HAS LNVINGFLG.
           $PCS4 OR IF CORE2 HAS VING AND CORE1 HAS NPOSFLG THEN NSTGO
                        EL OBJLIST(CORE2).
        HOUSED AT <NNVAR>:  <ANNVAR>:
                     IF CORE1 HASNT APOSFLG THEN CORE HAS LNVINGFLG OF
                        CORE2.
        HOUSED AT <ANNVAR>:  <LNVAR>:
           $PCS4 OR IF CORE2 HAS LNVINGGFLG THEN CORE1 HAS QPOSFLG OR
                        "A" OR "AN" OR "THE" OR TQUAN OR EACHEVERY.
* WN50: AN ADJECTIVE OCCURS TO THE RIGHT OF N ONLY IF IT HAS
*        MODIFIERS, OR IS IN THE SUBCLASS AINRN, OR THE HOST
*        IS AN INDEFINITE PRONOUN ("A STUDENT GOOD AT MATH", "STUDENTS
*        PRESENT", "SOMETHING GOOD").
WN50:  HOUSED AT <LA>, <RA>:  <LAR>;
                <QN>:  <QNPOS>:
                  CORE HAS CKADJINRNFLG.
        HOUSED AT <QNPOS>:  <LQNR>;
                <LQNR>, <LAR>:  <ADJINRN>;
                <ADJINRN>:  <REPRN>;
                <REPRN>:  <RN>:
                  CORE HAS CKADJINRNFLG OF CORE2.
        HOUSED AT <ADJINRN>:  <REPRN>:
                  CORE HAS ADJINRNFLG.
        HOUSED AT <REPRN>:  <RN>:
                  CORE HAS ADJINRNFLG OF CORE2.
        HOUSED AT <RN>:  <LNR>,1,2; <LNR>,2,2:
           $PCS4 OR IF CORE2 HAS ADJINRNFLG THEN CORE2 HAS
                        OKADJINRNFLG OR AINRN.
        HOUSED AT <RN>:  <LPRCR>:
           $PCS4 OR IF CORE2 HAS ADJINRNFLG THEN CORE2 HAS
```

```
                        OKADJINRNFLG OR AINRN OR CORE1 HAS
                        PRO:INDEFINITE.
* WN51:  RESTRICTION ON PN AS RIGHT ADJUNCT OF N:
*                 IF P  IS "INTO" THEN THE HOST IS NOT HUMAN (*"THE
*       CHILDREN INTO THE HOUSE ARE NOISY").  IF P IS "SINCE", THE HOST
*       IS A TIME NOUN ("THE TIME SINCE HIS ARRIVAL").  IF THE HOST IS A
*       NAME, THEN P IS "OF", "IN", "FROM" OR "AT".
WN51:  HOUSED AT <PN>:  <RNP>:
                  CORE HAS PNFLG.
       HOUSED AT <RNP>:  <REPRN>;
                 <REPRN>:  <RN>:
                  CORE HAS PNFLG OF CORE2.
       HOUSED AT <RN>:  <LNR>,1,2; <LPROR>:
          $PCS5 OR IF CORE2 HAS PNFLG AND "INTO" THEN CORE1 HASNT
                        NHUMAN.
          $PCS5 OR IF CORE2 HAS PNFLG AND  ("SINCE" OR "PRIOR TO"
                        OR "AWAY") THEN CORE1 HAS NTIME1.
          $PCS5 OR IF CORE2 HAS PNFLG AND  CORE1 HAS NAME THEN
                        CORE2 HAS "OF" OR "IN" OR "FROM" OR "AT",
```

```
*
*                        POSITION RESTRICTIONS
*
* DPCS1:  WHEN NOT A STRING IN SN, TOVO OCCURS AS OBJECTBE ONLY IN
*       "IS TO" CONSTRUCTION ("HE IS TO REPORT ON MONDAY").
DPCS1:  HOUSED AT <TOVO>:  <OBJECTBE>:
                    CORE HAS TOVOOBFLG.
        HOUSED AT <OBJECTBE>:  <OBJECT>;
                  <OBJECT>:  <YESNOQVO>; <TSUBJVOVO>:
                    CORE HAS TOVOOBFLG OF CORE2.
        HOUSED AT <OBJECT>:  <YESNOQVO>; <TSUBJVOVO>:
                    IF SPAN1 ISNT NL THEN CORE2 HASNT TOVOOBFLG.
        HOUSED AT <OBJECT>:  <VO>; <VINGO>; <VENO>; <TOVO>;
                             <ASSERTIONVO>;
                  <YESNOQVO>:  <YESNOQ>;
                  <TSUBJVOVO>:  <TSUBJVO>:
           $POS3 OR IF CORE2 HAS TOVOOBFLG THEN CORE1 HAS TV:VBE.
* DPCS2:  CS1 SUBCLASS CS1INNER CANNOT OCCUR IN SENTENCE INITIAL
*     POSITION (*"FOR HE HATED SCHOOL, HE JOINED THE ARMY").
DPCS2:  HOUSED AT <ASSERTION>:  <SUB1>:
                    IF CORE1 HAS CS1:CS1INNER THEN CORE HAS SAINNERFLG.
        HOUSED AT <SUB1>:  <CSSTG>;
                  <CSSTG>:  <REPSA>;
                  <REPSA>:  <SA>:
                    CORE HAS SAINNERFLG OF CORE2.
        HOUSED AT <SA>:  <ASSERTION>,1; <YESNOQ>,1; <WHQ>,1; <PWHQ>,1;
                         <PWHQ-PN>,1; <WHNQ-N>,1; <PWHNQ-PN>,1;
                         <PWHNQ>,1:
           $POS3 OR CORE2 HASNT SAINNERFLG.
* DPCS3:  NO APPOSITION IF THE SUBJECT HAS NOT OCCURRED.
DPCS3:  HOUSED AT <RNSUBJ>:  <REPSA>:
                    CORE HAS RNSUBJFLG.
        HOUSED AT <REPSA>:  <SA>;
                  <SA>:  <ASSERTIONVO>,5; <TOVO>,7; <VENPASS>,6;
                         <NULSASOBJBE>:
                    CORE HAS RNSUBJFLG OF CORE2.
        HOUSED AT <ASSERTIONVO>:  <ASSERTION>;
                  <TOVO>:  <NTOVO>;
                  <VENPASS>:  <SVEN>;
                  <NULSASOBJBE>:  <SASOBJBE>;
                  <SA>:  <SOBJBE>,4:
           $POS3 OR IF CORE2 HAS RNSUBJFLG THEN CORE1 HASNT
                         NULLSUBJFLG.
* DPCS4:  RNSUBJ ONLY OCCURS IN POST-OBJECT POSITION.
DPCS4:  HOUSED AT<SA>:  ALL OCCURRENCES EXCEPT <ASSERTIONVO>,5;
                         <YESNOQVO>,4; <TSUBJVOVO>,4; <VENPASS>,6;
                         <VINGO>,6; <TOVO>,7; <SOBJBE>,4;
                         <VINGOFN>,6; <NULSASOBJBE>; <OBJBESA>:
           $POS3 OR CORE2 HASNT RSUBJFLG.
* DPCS5:  THE SAWHICHSTG OCCURS ONLY IN POST OBJECT POSITION ("HE
*       RECOVERED, WHICH SURPRISED US").  THE COMPARATIVE STRING HEADED
*       BY "THAN" IS RARE EXCEPT IN POST OBJECT POSITION ("MORE WORKS
*       WERE ATTRIBUTED TO HIM THAN COULD BE EXPECTED").
```

```
DPCS5:  HOUSED AT !WHICH!:  <NULSAWHICHSTG>:
                  CORE HAS WHICHFLG.
        HOUSED AT !THAN!:  <NULSAWHICHSTG>:
                  CORE HAS THANFLG.
        HOUSED AT <NULSAWHICHSTG>:  <SAWHICHSTG>;
                  <SAWHICHSTG>:  <SAWH>;
                  <SAWH>:  <REPSA>;
                  <REPSA>:  <SA>:
                    CORE HAS WHICHFLG, THANFLG OF CORE2.
        HOUSED AT <SA>:  ALL OCCURRENCES EXCEPT <ASSERTIONVO>,5;
                         <YESNOQVO>,4; <TSUBJVOVO>,4; <VENPASS>,6;
                         <VINGO>,6; <TOVO>,7; <SOBJBE>,4;
                         <VINGOFN>,6; <NULSASOBJBE>; <OBJBESA>:
           $POS4 OR CORE2 HASNT WHICHFLG.
           $RARE OR CORE2 HASNT THANFLG.
* DPCS7:  CERTAIN OPTIONS OF SA OCCUR ONLY IN CENTER LIKE STRINGS.
DPCS7:  HOUSED AT <SAWHICHSTG>:  <SAWH>;
                  <NSTGT>, <CBJBESA>, <SOBJBESA>, <VINGO>, <VENPASS>,
                  <RNSUBJ>, <CSSTG>:  <REPSA>:
                    CORE HAS CENTERFLG.
        HOUSED AT <SAWH>:  <REPSA>;
                  <REPSA>:  <SA>;
                  <SA>: <VENPASS>,3; <TOVO>,4; <VINGO>,3; <OBJBESA>;
                        <VENPASS>,6; <TOVO>,7; <VINGO>,6:
                    CORE HAS CENTERFLG OF CORE2.
        HOUSED AT <SA>:  ALL OCCURRENCES EXCEPT <ASSERTION>;
                         <ASSERTIONVO>; <YESNOQ>; <YESNOQVO>;
                         <TSUBJVO>; <TSUBJVOVO>; <OBES>; FORTOVO>;
                         <NTOVO>; <SOBJBE>; <SASOBJBE>; <STOVO-N>;
                         <SOBJBESA>; <NVSA>; <VENPASS>; <TOVO>;
                         <VINGO>;
                  <SA>:  <VINGOFN>,6;
                  <OBJBESA>:  ALL OCCURRENCES EXCEPT <NULSOBJBESA>;
                  <TOVO>, <VINGO>:  <OBJECT>; <PASSOBJ>;
                  <VENPASS>:  <PASSOBJ>:
           $POS3 OR CORE2 HASNT CENTERFLG.
* DPCS10:  SENTENCE ADVERBS MAY PRECEDE THE VERB IN THE ADJUNCT (NOT OBJECT)
*          STRINGS VINGO AND VENPASS ("ESSENTIALLY FOLLOWING THE METHOD OF
*         LEVY, WE..."), IN "WE WERE ESSENTIALLY FOLLOWING THE METHOD
*         OF LEVY", THE SENTENCE ADVERB IS PART OF THE CENTER STRING.
DPCS10: HOUSED AT <LVSA>:  <VENPASS>; <VINGO>:
                    CORE HAS SAFLG.
        HOUSED AT <VENPASS>:  <OBJECT>; <PASSOBJ>; <OBJECTBE>:
           $POS3 OR CORE2 HASNT SAFLG.
* DPCS11:  ORNOT ONLY IF YESNOQ IS NOT PART OF ANOTHER STRING.
DPCS11: HOUSED AT <ORNOT>:  <YESNOQVO>:
                    CORE HAS ORNOTFLG.
        HOUSED AT <YESNOQVO>:  <YESNOQ>:
                    CORE HAS ORNOTFLG OF CORE2.
        HOUSED AT <YESNOQ>:  <NUL2WHQ-N>; <WHQ>; <PWHQ-PN>; <PWHQ>;
                             <WHNQ-N>; <PWHNQ-PN>; <PWHNQ>:
```

```
            $POS3 OR CORE2 HASNT ORNOTFLG.
* DPCS12:  SENTENCE ADJUNCT STRING NVSA ("I KNOW, PEOPLE THINK") CANNOT
*  OCCUR AT THE BEGINNING OF A SENTENCE.
DPCS12: HOUSED AT <NVSA>:  <REPSA>:
                     CORE HAS SAINNERFLG.
* DPCS14:  CERTAIN WH-STRINGS ARE RARE AS RNSUBJ.
DPCS14: HOUSED AT <WHENS>:  <RNWH>:
                     CORE HAS WHENSFLG.
        HOUSED AT <RNWH>:  <RNSUBJ>:
            $RARE OR CORE2 HAS WHENSFLG.
* DPCS15:  CERTAIN PREPOSITIONS DO NOT OCCUR IN RV.
DPCS15: HOUSED AT <PN>:  <REPRV>:
            $POS5 OR CORE2 HASNT !OF! OR !ABOUT! OR !SINCE! OR
                        !THROUGHOUT! OR !TO! OR !DURING!.
* DPCS16:  FOR WHETHS TO BE SENTENCE ADJUNCT, IT MUST START WITH "WHETHER".
DPCS16:  HOUSED AT <NULWHETHS>:  <WHETHS>:
                     IF CORE2 HAS !WHETHER! THEN CORE HAS WHETHERFLG.
        HOUSED AT <WHETHS>:  <SAWH>:
            $POS4 OR CORE2 HAS WHETHERFLG.
* DPCS17:  VALID ADVERB IN STRING OF ADVERBS.
DPCS17: HOUSED AT <*D>:  <DSTG>,2,1:
            $POS4 OR CORE2 HAS DSA OR DLP OR DVERY OR DLCOMP.
* DPCS18: ADJECTIVE OPTION OF OBJBE DOES NOT OCCUR IN SUB0 IF CS0 IS CS0AS
*       (SINCE, AS) ("AS A YOUNG MAN, HE LIKED SPORTS", *"AS YOUNG HE LIKED
*       SPORTS").
DPCS18: HOUSED AT <OBJBE>:  <SUB0>:
            $POS4 OR IF CORE2 HAS ASTGOBFLG THEN CORE1 HASNT CS0AS.
* WPCS1:  SUITABLE ADVERB FOR GIVEN SYNTACTIC POSITION ("HE RAN SWIFTLY",
*       *"HE RAN VERY").
WPCS1A: HOUSED AT <DSTG>:  <REPSA>:
            $POS4 OR CORE2 HAS DSA.
WPCS1B: HOUSED AT <DSTG> :  <REPRN>:
            $POS4 OR CORE2 HAS DRN.
WPCS1C: HOUSED AT <DSTG>:  <RW>:
            $POS4 OR CORE2 HAS DRW.
WPCS1D: HOUSED AT <DSTG>:  <LV>:
            $POS4 OR CORE2 HAS DLV.
WPCS1E: HOUSED AT <DSTG>:  <LPRO>:
            $POS4 OR CORE2 HAS DLTPRO.
WPCS1F: HOUSED AT <*D>:  <LCS>:
            $POS4 OR CORE2 HAS DLCS.
WPCS1G: HOUSED AT <*D>:  <LT>:
            $POS4 OR CORE2 HAS DLT OR DLTPRO.
WPCS1H: HOUSED AT <DSTG>:  <OBJBE>; <NULOBES>:
            $POS4 OR CORE2 HAS DPRED.
WPCS1I: HOUSED AT <*D>:  <LW>:
            $POS4 OR CORE2 HAS DLW.
WPCS1J: HOUSED AT <DSTG>:  <LQ>; <LQN>:
            $POS4 OR CORE2 HAS DLQ.
WPCS1K: HOUSED AT <DSTG>:  <LP>:
            $POS4 OR CORE2 HAS DLP.
```

```
WPCS1L: HOUSED AT <DSTG>:  <MULTSUBJVO>:
          $POS4 OR CORE2 HAS DPERM.
WPCS1M: HOUSED AT <DSTG>:  <MD>:
          $POS4 OR CORE2 HAS DLOC1.
WPCS1N: HOUSED AT <DSTG>:  <LVSA>:
          $POS4 OR CORE2 HAS DSA.
WPCS1O: HOUSED AT <*D>:  <SUBS>:
          $POS4 OR CORE2 HAS DUNIV.
WPCS1P: HOUSED AT <DSTG>:  <OBJECT>:
          $POS4 OR CORE2 HAS DEVAL.
* WPCS2:  IN RV, D IS DRV.  IF D IS ALSO DLOC3, THEN THE HOST IS A VERB
*      OF MOTION ("HE WALKED BY").
WPCS2:  HOUSED AT <DSTG>:  <REPRV>:
          $POS4 OR CORE2 HAS DRV.
        HOUSED AT <REPRV>:  <RV>:
                    IF CORE2 HAS DLOC3 THEN CORE HAS MOTIONFLG.
        HOUSED AT <RV>:  <RV>; <NTOVO>; <NTHATS>; <NSNWH>; <PNTHATS>;
                    <PNTHATSVO>; <PNSNWH>; <STOVO-N>;
                <NTOVO>, <NTHATS>, <NSNWH>, <PNTHATS>, <PNTHATSVO>,
                <PNSNWH>, <STOVO-N>:  <OBJECT>:
                  CORE HAS MOTIONFLG OF CORE2.
        HOUSED AT<RV>:  ALL OCCURRENCES EXCEPT <RV>; <NTOVO>;
                    <NTHATS>; <NSNWH>; <PNTHATS>; <PNTHATSVO>;
                    <PNSNWH>; <STOVO-N>;
                <OBJECT>:  ALL OCCURRENCES:
          $POS4 OR IF CORE2 HAS MOTIONFLG THEN CORE1 HAS VMOTION.
* WPCS3:  ADVERBS OCCURRING AS LA ARE IN ONE OF THE SUBCLASSES
*       DSA1 (GENERALLY LIKEABLE), DRV (POORLY WRITTEN), DLA (VERY TRUE)
*       DLCOMP (CONSIDERABLY BETTER).  HOWEVER AN ADVERB IN DRV SHOULD
*       NOT ALSO BE IN DLCC3 (A POORLY WRITTEN LETTER, *A BY WALKING
*       MAN).
WPCS3:  HOUSED AT <DSTG>:  <LA>:
          $POS4 OR CORE2 HAS DSA1 OR DRV OR DLA OR DLCOMP.
          $POS4 OR IF CORE2 HAS DRV THEN CORE2 HASNT DLCC3.
* WPCS4:  CORRECT ADVERB SUBCLASS TO THE LEFT OF A OR ADJECTIVAL VING/VEN,
*       AS OPPOSED TO ADVERB MODIFYING VERBAL VING/VEN.
WPCS4:  HOUSED AT <AVAR>:  <LAR>:
          $POS4 OR IF CORE1 HAS DRV AND NOT (DSA OR DLA) THEN CORE2
                    HAS VING OR VEN.
          $POS4 OR IF CORE1 HAS DLA AND NOT (DSA OR DRV) THEN CORE2
                    HAS ADJ OR VING:VVERYVING.
* WPCS5:  SUBJECT PRONOUN IS NOT ACCUSATIVE EXCEPT IN THE NOMINALIZATION
*       STRINGS FORTOVO, NTOVO, ETC. ("HE WENT", "FOR HIM TO GO").
WPCS5:  HOUSED AT <SUBJECT>:  <FORTOVO>; <NTOVO>; <STOVO-N>; <SOBJBE>;
                              <SVEN>; <SASOBJBE>;
          $SS  OR  CORE2 HASNT PRO:NOMINATIVE.
        HOUSED AT <SUBJECT>:  <YESNOQ>; <TSUBJVO>; <OBES>:
          $SS  OR  CORE2 HASNT PRO:ACCUSATIVE.
        HOUSED AT <SUBJECT>:  <ASSERTION>:
                    IF CORE2 HAS PRO:NOMINATIVE THEN CORE HAS
                      PRONOMFLG.
                    IF CORE2 HAS PRO:ACCUSATIVE THEN CORE HAS PROACCFLG.
        HOUSED AT <ASSERTION>:  <SVO>:
```

```
            $SS  OR  CORE2 HASNT PRONOMFLG.
         HOUSED AT <ASSERTION>:  ALL OCCURRENCES EXCEPT <SVO>:
            $SS  OR  CORE2 HASNT PROACCFLG.
* WPCS6:  PRONOUN IS NOT NOMINATIVE WHEN OCCURRING IN THE POSITION OF AN
*       OBJECT NOUN ("I MET HIM", *"I MET HE").
WPCS6:  HOUSED AT <NSTG>:  <NSTGO>:
            $SS  OR  IF CORE2 HAS PRO THEN CORE2 HASNT NOMINATIVE.
* WPCS7:  THE ONLY TYPE PRONOUN WHICH OCCURS AS RSUBJ (RCVING ADJUNCT
*       OF SUBJECT) IS THE SUBCLASS PROSELF ("HE IS HIMSELF INTERESTED
*       IN THE PROBLEM").
WPCS7:  HOUSED AT <*PRO>:  <RSUBJ>:
            $POS4 OR CORE2 HAS PROSELF.
* WPCS8:  IN SUB0, IF OBJBE IS AN INDEFINITE PRONOUN, IT HAS A RIGHT
*        ADJUNCT ("AS SOMEONE INTERESTED IN THE PROBLEM, HE...",
*       *"AS SOMEONE, HE...").
WPCS8:  HOUSED AT <RN>:  <LPROR>:
                     CORE HAS RNFLG,
         HOUSED AT <LPROR>:  <NSTG>;
                   <NSTG>:  <OBJBE>:
                     CORE HAS RNFLG OF CORE2,
         HOUSED AT <OBJBE>:  <SUB0>:
            $POS3 OR CORE2 HAS RNFLG.
* WPCS9:  OBJBE CANNOT BE PRONOUN IN OBJBESA.
WPCS9:  HOUSED AT <OBJBE>:  <OBJBESA>:
            $POS3 OR CORE2 HASNT PRO.
* WPCS10:  A NOUN IN THE TIME PHRASE NSTGT OCCURRING AS SENTENCE ADJUNCT
*       (ONE DAY, LAST WEEK) MUST EITHER BE NTIME2 (E.G. YESTERDAY) OR ELSE
*       NTIME1 WITH APPROPRIATE LEFT OR RIGHT ADJUNCTS (AN HOUR AGO).
WPCS10: HOUSED AT <QPCS>:  <TQPOS>:
                     IF CORE1 HAS T:TIMETAG OR CORE2 HAS Q:TIMETAG THEN
                          CORE HAS TIMETAGFLG,
         HOUSED AT <APCS>:  <ANNVAR>:
                     IF CORE2 HAS ADJ:TIMETAG THEN CORE HAS
                          TIMETAGFLG,
         HOUSED AT <TQPCS>, <ANNVAR>:  <LNVAR>;
                   <LNVAR>:  <LNR>:
                     CORE HAS TIMETAGFLG OF CORE2,
         HOUSED AT <RN>:  <LNR>,1,2:
                     IF CORE2 HAS D:TIMETAG THEN CORE HAS TIMETAGFLG,
         HOUSED AT <LNR>: <NSTGT>:
            $POS4 OR CORE2 HAS NTIME2 OR (NTIME1 AND TIMETAGFLG).
* WPCS11:  IN PA, A IS AINPA; P IS "IN" OR "AT" (IN GENERAL, AT BEST).
WPCS11: HOUSED AT <*ADJ>:  <PA>:
            $POS4 OR CORE2 HAS AINPA.
         HOUSED AT <*P>:  <PA>:
            $POS5 OR CORE2 HAS !IN! OR !AT!.
* WPCS12:  IN NQ, THE NOUN IS NPREQ OR NAME ("A MODEL 6 RADIO").
WPCS12: HOUSED AT <*N>:  <NQ>:
            $POS4 OR CORE2 HAS NPREQ OR NAME.
* WPCS13:  A VING OR VEN ADJECTIVAL OBJBE IS IN SUBCLASS VVERYVING ("HE IS
*       (VERY) ENCOURAGING", CONTRAST: "HE IS ENCOURAGING US.").
WPCS13: HOUSED AT <ASTG>:  <OBJBE>:
            $POS4 OR IF CORE2 HAS VING THEN CORE2 HAS VVERYVING.
```

```
* WPCS14:  VEN AS ADJECTIVE HAS PASSIVE OBJECT LIST ("THEY HAVE HIRED
*     INSTRUCTORS" ("HIRED" AS LN).  *"THEY HAVE BEEN INSTRUCTORS"  ("BEEN"
*     AS LN)),
WPCS14: HOUSED AT <*VEN>:  <NULAVAR>:
          $POS4 OR CORE2 HAS POBJLIST
* WPCS15:  P IN P OBJECT STRING MUST BE THE ONE NAMED  AS ATTRIBUTE IN VERB;
*       SAME FOR DP OBJECT STRINGS.
WPCS15: HOUSED AT <PN>, <PVINGSTG>, <PSVINGO>, <PVINGO>, <PSNWH>,
                  <PSTG>, <DP1PN>:  <OBJECT>: <PASSOBJ>;
                  <PNTHATS>, <PNTHATSVO>, <PNSNWH>, <PNN>,
                  <PNVINGSTG>, <NPN>, <VINGSTGPN>, <NPVINGSTG>,
                  <NPSVINGO>, <NPVINGO>, <NPSNWH>, <DP2PN>,
                  <DP3PN>, <DP4PN>:  <OBJECT>;
                  <DP1PN>:  <PASSOBJ>:
                    CORE HAS PCBJFLG.
        HOUSED AT <DP1>, <DP1PN>:  <OBJECT>; <PASSOBJ>;
                  <DP2>, <DP3>, <DP4>, <DPSN>, <DP2PN>, <DP3PN>,
                  <DP4PN>:  <OBJECT>;
                  <DP1P>:  <PASSOBJ>:
                    CORE HAS DPOBJFLG.
        HOUSED AT <NULOPPN>:  <DP1PN>, <DP2PN>, <DP3PN>, <DP4PN>:
                  <*P>:  <DP1P>:
                    IF CORE1 HAS !ABOUT! THEN CORE HAS ABOUTFLG.
                    IF CORE1 HAS !ACROSS! THEN CORE HAS ACROSSFLG.
                    IF CORE1 HAS !ALONG! THEN CORE HAS ALONGFLG.
                    IF CORE1 HAS !AROUND! THEN CORE HAS AROUNDFLG.
                    IF CORE1 HAS !AWAY! THEN CORE HAS AWAYFLG.
                    IF CORE1 HAS !BACK! THEN CORE HAS BACKFLG.
                    IF CORE1 HAS !DOWN! THEN CORE HAS DOWNFLG.
                    IF CORE1 HAS !IN! THEN CORE HAS INFLG.
                    IF CORE1 HAS !OFF! THEN CORE HAS OFFFLG.
                    IF CORE1 HAS !ON! THEN CORE HAS ONFLG.
                    IF CORE1 HAS !OUT! THEN CORE HAS OUTFLG.
                    IF CORE1 HAS !OVER! THEN CORE HAS OVERFLG.
                    IF CORE1 HAS !THROUGH! THEN CORE HAS THROUGHFLG.
                    IF CORE1 HAS !UP! THEN CORE HAS UPFLG.
        HOUSED AT <OBJECT>:  <YESNOQVOVO>; <TSUBJVOVO>; <ASSERTIONVO>;
                             <VENO>; <VINGO>; <VO>; <TOVO>; <VINGOFN>:
           $POS5 OR IF CORE2 HAS POBJFLG THEN CORE1 HAS CBJLIST:PN
                       AND CORE2 EL OBJLIST:PN:PVAL(CORE1),
           $POS5 OR IF CORE2 HAS DPOBJFLG AND NOT POBJFLG THEN CORE1
                       HAS OBJLIST:DP2 AND CORE2 EL OBJLIST:DP2:DPVAL
                       (CORE1),
           $POS5 OR IF CORE2 HAS DPOBJFLG AND POBJFLG AND ABOUTFLG
                       THEN CORE1 HAS OBJLIST:DP2 AND !ABOUT!
                       EL CBJLIST:DP2:DPVAL(CORE1),
           $POS5 OR IF CORE2 HAS DPOBJFLG AND POBJFLG AND ACROSSFLG
                       THEN CORE1 HAS OBJLIST:DP2 AND !ACROSS!
                       EL CBJLIST:DP2:DPVAL(CORE1),
           $POS5 OR IF CORE2 HAS DPOBJFLG AND POBJFLG AND ALONGFLG
                       THEN CORE1 HAS OBJLIST:DP2 AND !ALONG!
                       EL CBJLIST:DP2:DPVAL(CORE1),
           $POS5 OR IF CORE2 HAS DPOBJFLG AND POBJFLG AND AROUNDFLG
```

```
                    THEN CORE1 HAS OBJLIST:DP2 AND !AROUND!
                    EL CBJLIST:DP2:DPVAL(CORE1).
    $POS5 OR IF CORE2 HAS DPOBJFLG AND POBJFLG AND AWAYFLG
                    THEN CORE1 HAS OBJLIST:DP2 AND !AWAY!
                    EL CBJLIST:DP2:DPVAL(CORE1).
    $POS5 OR IF CORE2 HAS DPOBJFLG AND POBJFLG AND BACKFLG
                    THEN CORE1 HAS OBJLIST:DP2 AND !BACK!
                    EL CBJLIST:DP2:DPVAL(CORE1).
    $POS5 OR IF CORE2 HAS DPOBJFLG AND POBJFLG AND DOWNFLG
                    THEN CORE1 HAS OBJLIST:DP2 AND !DOWN!
                    EL CBJLIST:DP2:DPVAL(CORE1).
    $POS5 OR IF CORE2 HAS DPOBJFLG AND POBJFLG AND INFLG
                    THEN CORE1 HAS OBJLIST:DP2 AND !IN!
                    EL CBJLIST:DP2:DPVAL(CORE1).
    $POS5 OR IF CORE2 HAS DPOBJFLG AND POBJFLG AND OFFFLG
                    THEN CORE1 HAS OBJLIST:DP2 AND !OFF!
                    EL CBJLIST:DP2:DPVAL(CORE1).
    $POS5 OR IF CORE2 HAS DPOBJFLG AND POBJFLG AND ONFLG
                    THEN CORE1 HAS OBJLIST:DP2 AND !ON!
                    EL CBJLIST:DP2:DPVAL(CORE1).
    $POS5 OR IF CORE2 HAS DPOBJFLG AND POBJFLG AND OUTFLG
                    THEN CORE1 HAS OBJLIST:DP2 AND !OUT!
                    EL CBJLIST:DP2:DPVAL(CORE1).
    $POS5 OR IF CORE2 HAS DPOBJFLG AND POBJFLG AND OVERFLG
                    THEN CORE1 HAS OBJLIST:DP2 AND !OVER!
                    EL CBJLIST:DP2:DPVAL(CORE1).
    $POS5 OR IF CORE2 HAS DPOBJFLG AND POBJFLG AND THROUGHFLG
                    THEN CORE1 HAS OBJLIST:DP2 AND !THROUGH!
                    EL CBJLIST:DP2:DPVAL(CORE1).
    $POS5 OR IF CORE2 HAS DPOBJFLG AND POBJFLG AND UPFLG
                    THEN CORE1 HAS OBJLIST:DP2 AND !UP!
                    EL CBJLIST:DP2:DPVAL(CORE1).
HOUSED AT <PASSOBJ>:  <VENPASS>:
    $POS5 OR IF CORE2 HAS POBJFLG THEN CORE1 HAS POBJLIST:PN
                    AND CORE2 EL POBJLIST:PN:PVAL(CCRE1).
    $POS5 OR IF CORE2 HAS DPOBJFLG AND NOT POBJFLG THEN CORE1
                    HAS POBJLIST:DP2 AND CORE2 EL
                    POBJLIST:DP2:DPVAL(CORE1).
    $POS5 OR IF CORE2 HAS DPOBJFLG AND POBJFLG AND ABOUTFLG
                    THEN CORE1 HAS POBJLIST:DP2 AND !ABOUT!
                    EL POBJLIST:DP2:DPVAL(CORE1).
    $POS5 OR IF CORE2 HAS DPOBJFLG AND POBJFLG AND ACROSSFLG
                    THEN CORE1 HAS POBJLIST:DP2 AND !ACROSS!
                    EL POBJLIST:DP2:DPVAL(CORE1).
    $POS5 OR IF CORE2 HAS DPOBJFLG AND POBJFLG AND ALONGFLG
                    THEN CORE1 HAS POBJLIST:DP2 AND !ALONG!
                    EL POBJLIST:DP2:DPVAL(CORE1).
    $POS5 OR IF CORE2 HAS DPOBJFLG AND POBJFLG AND AROUNDFLG
                    THEN CORE1 HAS POBJLIST:DP2 AND !AROUND!
                    EL POBJLIST:DP2:DPVAL(CORE1).
    $POS5 OR IF CORE2 HAS DPOBJFLG AND POBJFLG AND AWAYFLG
                    THEN CORE1 HAS POBJLIST:DP2 AND !AWAY!
                    EL POBJLIST:DP2:DPVAL(CORE1).
```

```
            $POSS OR IF CORE2 HAS DPOBJFLG AND POBJFLG AND BACKFLG
                        THEN CORE1 HAS POBJLIST:DP2 AND !BACK!
                        EL POBJLIST:DP2:DPVAL(CORE1).
            $POSS OR IF CORE2 HAS DPOBJFLG AND POBJFLG AND DOWNFLG
                        THEN CORE1 HAS POBJLIST:DP2 AND !DOWN!
                        EL POBJLIST:DP2:DPVAL(CORE1).
            $POSS OR IF CORE2 HAS DPOBJFLG AND POBJFLG AND INFLG
                        THEN CORE1 HAS POBJLIST:DP2 AND !IN!
                        EL POBJLIST:DP2:DPVAL(CORE1).
            $POSS OR IF CORE2 HAS DPOBJFLG AND POBJFLG AND OFFFLG
                        THEN CORE1 HAS POBJLIST:DP2 AND !OFF!
                        EL POBJLIST:DP2:DPVAL(CORE1).
            $POSS OR IF CORE2 HAS DPOBJFLG AND POBJFLG AND ONFLG
                        THEN CORE1 HAS POBJLIST:DP2 AND !ON!
                        EL POBJLIST:DP2:DPVAL(CORE1).
            $POSS OR IF CORE2 HAS DPOBJFLG AND POBJFLG AND OUTFLG
                        THEN CORE1 HAS POBJLIST:DP2 AND !OUT!
                        EL POBJLIST:DP2:DPVAL(CORE1).
            $POSS OR IF CORE2 HAS DPOBJFLG AND POBJFLG AND OVERFLG
                        THEN CORE1 HAS POBJLIST:DP2 AND !OVER!
                        EL POBJLIST:DP2:DPVAL(CORE1).
            $POSS OR IF CORE2 HAS DPOBJFLG AND POBJFLG AND THROUGHFLG
                        THEN CORE1 HAS  OBJLIST:DP2 AND !THROUGH!
                        EL POBJLIST:DP2:PPVAL(CORE1).
            $POSS OR IF CORE2 HAS DPOBJFLG AND POBJFLG AND UPFLG
                        THEN CORE1 HAS POBJLIST:DP2 AND !UP!
                        EL POBJLIST:DP2:DPVAL(CORE1).
* WPCS18:  PN FOLLOWING VERB IS NOT TAKEN AS RV IF IT IS ELIGIBLE FOR PN
*        OBJECT OF VERB UNLESS RARE SWITCH IS ON, ("HE IDENTIFIED WITH
*        IMMEDIACY WITH JOHN"),  SIMILARLY FOR POST OBJECT RV AND NPN
*        ("HE COMPARED THE SAMPLES WITH THE PREVIOUS LOT").
WPCS18: HOUSED AT <PN>:  <REPRV>:
                  CORE HAS PNFLG,
        HOUSED AT <REPRV>:  <RV>:
                  CORE HAS PNFLG OF CORE2,
        HOUSED AT <RV>:  <VENPASS>:
           $RARE OR IF CORE2 HAS PNFLG AND CORE1 HAS DVBFLG AND NOT
                        NULLOBJFLG THEN CORE2 NEL
                        POBJLIST:NPN:PVAL(CORE1),
           $RARE OR IF CORE2 HAS PNFLG AND CORE1 HAS NULLOBJFLG THEN
                        CORE2 NEL POBJLIST:PN:PVAL(CORE1),
        HOUSED AT <RV>:  <ASSERTIONVO>; <TSUBJVOVO>; <VINGOFN>; <VENC>;
                        <VINGO>; <VC>; <TOVO>; <YESNOQVC>:
           $RARE OR IF CORE2 HAS PNFLG AND CORE1 HAS NULLOBJFLG THEN
                        CORE2 NEL OBJLIST:PN:PVAL(CORE1),
           $RARE OR IF CORE2 HAS PNFLG AND CORE1 HAS DVBFLG AND NOT
                        NULLOBJFLG THEN CORE2 NEL
                        OBJLIST:NPN:PVAL(CORE1),
        HOUSED AT <RV>:  <LTVR>; <LVR>; <LVENR>; <LVINGR>:
           $RARE OR IF CORE2 HAS PNFLG THEN CORE2 NEL
                        OBJLIST:PN:PVAL(CORE1),
        HOUSED AT <VERB>:  <ASSERTIONVO>;
                  <VERB2>:  <TSUBJVOVO>; <YESNOQVO>;
```

```
                    <*VING>:  <VINGOFN>;
                    <LVENR>:  <VENO>; <VENPASS>;
                    <LVINGR>:  <VINGO>;
                    <LVR>:  <VC>; <TOVC>:
                      CORE HAS NULLOBJFLG.
          HOUSED AT <OBJECT>:  <ASSERTIONVO>; <TSUBJVOVC>; <VINGOFN>;
                                 <VENO>; <VINGO>; <VO>; <TOVO>;
                    <PASSOBJ>:  <VENPASS>:
                      CORE HASNT NULLOBJFLG.
* WPCS19:  ONLY BIG SA'S IN PRCSENT.
WPCS19: HOUSED AT <SAWHICHSTG>:  <SAWH>;
                    <*INT>, <PN>, <PA>, <NSTGT>, <RSUBJ>,
                    <RNSUBJ>:  <REPSA>:
                      CORE HAS NOPROFLG.
          HOUSED AT <SAWH>:  <REPSA>;
                    <REPSA>:  <SA>:
                      CORE HAS NOPROFLG OF CORE2.
          HOUSED AT <SA>:  <PRCSENT>:
             $POS3 OR CORE2 HASNT NOPRCFLG.
* WPCS20:  NAMESTG ENDS IN A NAME OR SYMBOL.
WPCS20: HOUSED AT <*N>:  <NULNAMESTG>:
             $POS4 OR CORE2 HAS NAME OR NSYMBOL.
* WPCS21:  NOUN POSITION OF LEFT ADJUNCT OF NOUN CANNOT BE NAME OR SYMBOL.
WPCS21: HOUSED AT <NVAR>:  <NNVAR>:
             $POS4 OR IF CORE1 HAS NPOSFLG THEN CORE2 HASNT NAME OR
                          NSYMBOL.
* WPCS22:  A GROUP OF VERBS VNN1 (MAKE, SHOW) WHICH TAKE NN AS OBJECT
*      REQUIRE THE FIRST NOUN TO BE HUMAN ("SHOW HIM A BOOK").
WPCS22: HOUSED AT <NSTGO>:  <NN>,1:
                      IF CORE2 HASNT N:NHUMAN OR N:AGGREGATE OR
                           PRO:NHUMAN OR PRO:PLURAL THEN CORE HAS
                           VMODFLG.
          HOUSED AT <NN>:  <OBJECT>:
                      CORE HAS NNFLG.
                      CORE HAS VMODFLG OF CORE2.
          HOUSED AT <OBJECT>:  <ASSERTIONVO>;<TSUBJVOVO>; <VINGOFN>;
                                 <VENO>; <VINGO>; <VO>; <TOVC>; <YESNOQVC>;
             $POS4 OR IF CORE2 HAS NNFLG AND VMODFLG THEN CORE1 HAS VMOD.
* WPCS23:  QUESTION IS FOLLOWED BY QUESTION MARK.  QUESTION MARK IS
*      PRECEDED BY QUESTION.
WPCS23: HOUSED AT <QUESTION>:  <CENTER>:
                      CORE HAS QUESTIONFLG.
          HOUSED AT <CENTER>:  <SENTENCE>:
                      CORE HAS QUESTIONFLG OF CORE2.
          HOUSED AT <ENDMARK>:  <SENTENCE>:
             $PUNC OR IF CORE1 HAS QUESTIONFLG THEN CORE2 HAS !?!.
             $PUNC OR IF CORE2 HAS !?! THEN CORE1 HAS QUESTIONFLG.
* WPCS24:  VEN BANNED FROM ADJECTIVE ALTERNATIVE OF OBJECT.
WPCS24: HOUSED AT <AVAR>:  <LAR>:
             $POS3 OR CORE2 HASNT VEN.
* WPCS25:  CERTAIN POSITIONS CANNOT HAVE LEFT ADJUNCT WHEN THEY
*      OCCUR IN RV OR SA.
WPCS25: HOUSED AT <PN>:  <REPSA>:
```

```
   $POS5 OR CCRE2 HASNT LPFLG AND !OF!.
HOUSED AT <PN>!  <REPRV>!
   $POS5 OR CCRE2 HASNT LPFLG AND (!OF! OR !ABOUT! OR !SINCE!
                 OR !THROUGHOUT! OR !TO! OR !DURING!).
```

```
*
*                    QUANTITIES RESTRICTIONS
*
* DQ1:  THE DIMENSIONAL SEQUENCE QN DOES NOT OCCUR AS A RIGHT ADJUNCT OF
*       N UNLESS THERE IS AN A OR PN IN QN ("A PORTICO 100 FEET LONG WAS
*       SEEN", *"A PORTICO 100 FEET WAS SEEN").
DQ1:  HOUSED AT <SCALESTG>:  <QN>:
                  CORE HAS OKADJINRNFLG.
      HOUSED AT <QN>:  <QNPOS>;
                <QNPOS>:  <LQNR>:
                  CORE HAS OKADJINRNFLG OF CORE2.
      HOUSED AT <LQNR>:  <ADJINRN>:
         $POS3 OR CORE2 HAS OKADJINRNFLG.
* DQ2: QN DOES NOT CONTAIN PN IF IT IS OCCURRING IN LN (*"A THREE INCH
*       IN LENGTH LINE").
DQ2:  HOUSED AT <PN>:  <SCALESTG>:
                  CORE HAS PNFLG.
      HOUSED AT <SCALESTG>:  <QN>;
                <QN>:  <QNPOS>;
                <QNPOS>:  <LQNR>:
                  CORE HAS PNFLG OF CORE2.
      HOUSED AT <LQNR>:  <ADJADJ>:
         $POS3 OR CORE2 HASNT PNFLG.
* DQ3:  THE QN STRING IN ADJECTIVAL POSITION CAN BE FOLLOWED BY ADJ
*       ONLY IF N IN QN IS NUNITS ("A TWO INCH LONG LINE", *"A TWO
*       FAMILY LARGE HOUSE").  IN RV AND LP QN DOES NOT OCCUR WITH ADJ
*       OR PN (*"HE THREW IT FIVE FEET LONG").
DQ3:  HOUSED AT <*ADJ>:  <SCALESTG>:
                  CORE HAS ADJFLG.
      HOUSED AT <SCALESTG>:  <QN>;
                <QN>:  <QNPOS>:
                  CORE HAS ADJFLG OF CORE2.
      HOUSED AT <QN>: <REPRV>;
                <QNPOS>:  <LP>:
         $POS3 OR CORE2 HASNT PNFLG OR ADJFLG.
      HOUSED AT <SCALESTG>:  <QN>:
         $POS4 OR IF CORE2 HAS ADJFLG OR PNFLG THEN CORE1 HAS NUNIT.
* WQ1:  NUMBER IN MEASURE SEQUENCE QN: IN LN, N IS SINGULAR; IN
*       PREDICATE, IN RN, Q AND N AGREE IN NUMBER ("A TWO INCH LINE",
*       "IT IS TWO INCHES").
WQ1:  HOUSED AT <CPDNUMBR>:  <NUMBRSTG>:
                  CORE HAS PLURALFLG.
      HOUSED AT <NUMBRSTG>:  <QN>:
                  CORE HAS PLURALFLG OF CORE2.
      HOUSED AT <*N>:  <QN>:
                  IF NOT (IF CORE2 HAS SINGULAR THEN CORE1 HASNT
                       PLURAL OR PLURALFLG) AND NOT (IF CORE2 HAS
                       PLURAL THEN CORE1 HASNT SINGULAR) THEN CORE
                       HAS ONLYAPOSFLG.
                  IF CORE2 HAS PLURAL THEN CORE HAS NOAPOSFLG.
      HOUSED AT <QN>:  <QNPOS>;
                <QNPOS>:  <LQNR>:
                  CORE HAS ONLYAPOSFLG, NOAPOSFLG OF CORE2.
```

```
      HOUSED AT <QN>:  ALL OCCURRENCES EXCEPT <QNPOS>;
                <QNPOS>:  ALL OCCURRENCES EXCEPT <LQNR>;
                <LQNR>:  ALL OCCURRENCES EXCEPT <ADJADJ>:
         $SS  OR  CORE2 HASNT ONLYAPOSFLG.
      HOUSED AT <LQNR>:  <APOS>:
         $SS  OR  CORE2 HASNT NOAPOSFLG.
* WQ2:  IN QN, IF SCALESTG IS A, THEN A IS ASCALE (LONG, WIDE) OR
*       COMPARATIVE (LARGER) ("TWO INCHES LONG OR WIDE", "A SHADE DARKER
*       OR LIGHTER").
WQ2:  HOUSED AT <*ADJ>:  <SCALESTG>:
         $POS4 OR CORE2 HAS ASCALE OR COMPARATIVE.
* WQ3:  IN PN OPTION OF SCALESTG, N IS NSCALE ("TWO INCHES IN LENGTH")
WQ3:  HOUSED AT <NSTGO>:  <PN>:
                  IF CORE2 HAS NSCALE THEN CORE HAS SCALEFLG.
      HOUSED AT <PN>:  <SCALESTG>:
         $POS4 OR CORE2 HAS SCALEFLG.
* WQ4:  AMONG QUANTIFIERS, ONLY NUMBERS REPEAT, EXCEPT THAT THE LAST
*       QUANTIFIER IN A SEQUENCE OF REPEATED QUANTIFIERS MAY BE
*       COMPARATIVE MARKER (MORE, LESS) OR THE  WORD !SUCH! ("THREE
*       HUNDRED MORE"; "THREE HUNDRED SUCH").
WQ4:  HOUSED AT <*Q>:  <CPDNUMBR>,1,2; <CPDNUMBR>,2,2:
         $POS4 OR CORE2 HAS QNUMBER OR COMPARATIVE OR !SUCH! AND CORE1
                     HAS QNUMBER.
* WQ5:  Q IN NUMBRSTG MUST BE QNUMBER.
WQ5:  HOUSED AT <*Q>:  <NUMBRSTG>:
         $POS4 OR CORE2 HAS QNUMBER.
* WQ6:  ONLY THOSE ARTICLES AND QUANTIFIERS WHICH
*       ARE IN THE SUBCLASS QROVING (BOTH, ALL) OCCUR AS RSUBJ
*       (ROVING ADJUNCTS OF SUBJECT). IF Q OR T
*       IS FOLLOWED BY PN, THEN P = "OF" ("THEY ARE ALL OF THEM AGREED",
*       "THEY ARE EACH A GOOD THEORY").
WQ6:  HOUSED AT <*Q>, <*T>:  <NUL1RSUBJ>;
         $POS4 OR CORE2 HAS QROVING.
      HOUSED AT <PN>:  <NUL2RSUBJ>:
         $POS5 OR CORE2 HAS !OF!.
* WQ7:  QN OPTION OF LP CAN BE CHOSEN ONLY WITH APPROPRIATE PREPOSITION.
WQ7:  HOUSED AT <QNPOS>:  <LP>:
                  CORE HAS QNPOSFLG.
      HOUSED AT <LP>:  <PN>:
                  CORE HAS QNPOSFLG OF CORE2.
      HOUSED AT <*P>:  <PN>:
         $POS5 OR IF CORE1 HAS QNPOSFLG THEN CORE2 HAS !OFF! OR !INTO!
                     OR !OVER! OR !BELOW! OR !AFTER! OR !SINCE! OR
                     !FROM! OR !DOWN! OR !UNDER! OR !ABOVE!.
```

```
*
*                    SELECTION RESTRICTIONS
*
* DSEL1:  IF APOS IS VING, CCRE N IS SUITABLE SUBJECT FOR VING
DSEL1:  HOUSED AT <NNVAR>:  <ANNVAR>:
           $SEL1 OR IF CORE1 HAS VING AND CORE2 HAS N THEN
                        NOTNSUBJ(CCRE1)*CORE2=NL.
* WSEL1:  SUITABLE OBJECT N FOR GIVEN VERB.
WSEL1:  HOUSED AT <NSTGO>:  <OBJECT>:
                     CCRE HAS NSTGOFLG.
        HOUSED AT <OBJECT>:  <ASSERTIONVO>; <TSUBJVOVC>; <VINGOFN>;
                             <VENO>; <VINGO>; <VO>; <TOVO>; <YESNOQVC>:
           $SEL1 OR IF CORE2 HAS NSTGOFLG AND (N OR PRO) AND CORE1 HAS
                        NOTNOBJ THEN NOTNOBJ(CORE1)*CORE2=NL.
* WSEL2:  SUITABLE SUBJECT N FOR GIVEN VERB.
WSEL2:  HOUSED AT <ASSERTIONVO>:  <ASSERTION>;
                  <YESNOQVO>:  <YESNOQ>;
                  <TSUBJVCVO>:   <TSUBJVO>;
                  <TOVC>:  <FORTOVO>; <NTOVO>;
                  <VINGO>:  <SVINGO>;
                  <TOVO=N>:  <STOVO=N>:
           $SEL1 OR IF CORE1 HAS N OR PRO AND CORE2 HAS DVBFLG AND
                        NOTNSUBJ AND NOT DVBVENPASSFLG THEN
                        NOTNSUBJ(CORE2) * CORE1 = NL.
           $SEL1 OR IF CORE1 HAS !THERE! THEN CORE2 HAS DOBFLG.
* WSEL3:  IN COMPOUND NOUNS NVING (HAT WEARING) THE MOST PRODUCTIVE SUBTYPE
*       IS ONE WHERE THE NOUN IS A POSSIBLE OBJECT OF THE VING VERB.
WSEL3:  HOUSED AT <NVAR>:  <NNVAR>:
           $SEL1 OR IF CORE2 HAS VING AND NOTNOBJ AND CORE1 HAS N
                        THEN NOTNOBJ(CORE2) * CORE1 = NL.
* WSEL4:  SIMILAR TO WSEL3, EXCEPT IN ADJECTIVE POSITION ("A HAT WEARING
*       PERSON", *"AN EVIDENCE LEADING MACHINE").
WSEL4:  HOUSED AT <NULAVAR>:  <AVAR>;
           $SEL1 OR IF CORE1 HAS N AND CORE2 HAS VING AND NOTNOBJ
                        THEN NOTNOBJ(CORE2) * CORE1 = NL.
* WSEL5:  IF RN = VENPASS, THEN THE HOST NOUN IS NOT ON THE FORBIDDEN
*      OBJECT LIST OF THE VERB (IF "HE QUOTED A VERSE" THEN "THE VERSE
*      QUOTED").
WSEL5:  HOUSED AT <VENPASS>:  <REPRN>:
                     CORE HAS VENPASSFLG.
        HOUSED AT <REPRN>:  <RN>:
                     CORE HAS VENPASSFLG OF CORE2.
        HOUSED AT <RN>:  <LNR>,1,2; <LPROR>:
           $SEL1 OR IF CORE2 HAS VENPASSFLG AND NOTNOBJ AND CORE1 HAS
                        N THEN NOTNOBJ(CORE2) * CORE1 = NL.
```

```
*
*              SENTENCE NOMINALIZATION RESTRICTIONS
*
* DSN1: CORRECT SUBJECT FOR SN OR ASSERTION OBJECT OF !BE!
*       ("THE TRUTH IS WE NEED MONEY").
DSN1:   HOUSED AT <SN>, <ASSERTION>:  <OBJECTBE>:
                    CORE HAS SNACBFLG.
        HOUSED AT <FORTOVC>: <SN>:
                    CORE HAS FORTOVCFLG.
        HOUSED AT <TOVC>:  <SN>:
                    CORE HAS TOVCFLG.
        HOUSED AT <SN>:  <SUBJECT>; <OBJECTBE>:
                    CORE HAS FORTOVCFLG, TOVCFLG OF CORE2.
        HOUSED AT <OBJECTBE>:  <OBJECT>;
                  <OBJECT>:  <ASSERTIONVO>; <YESNOQVO>; <TSUBJVOVO>;
                             <VINGO>; <VENO>; <TOVO>; <VO>;
                  <PASSOBJ>:  <VENPASS>;
                  <VINGO>, <VENO>, <VO>, <VENPASS>, <TOVO>:  <OBJECT>;
                                                      <PASSOBJ>;
                  <TOVO>:  <TOVO-N>;
                  <VENPASS>:  <NULOBES>;
                  <NULOBES>:  <OBES>:
                    CORE HAS SNACBFLG,FORTOVCFLG,TOVCFLG OF CORE2.
        HOUSED AT <LN1>:  <LNR>,2,1:
                    CORE HAS NULLNFLG.
        HOUSED AT <LNR>:  <NSTG>;
                  <NSTG>:  <SUBJECT>;
                    CORE HAS NULLNFLG OF CORE2.
        HOUSED AT <SUBJECT>:  <ASSERTION>; <YESNOQ>; <TSUBJVO>;
                              <FORTOVO>; <NTOVO>; <STOVO-N>;
                              <SVEN>:
                    CORE HAS NULLNFLG, FORTOVCFLG, TOVCFLG OF CORE2.
        HOUSED AT <ASSERTIONVC>:  <ASSERTION>;
                  <YESNOQVO>:  <YESNOQ>;
                  <TSUBJVOVO>:  <TSUBJVO>;
                  <TOVC>:  <FORTOVO>; <NTOVO>;
                  <TOVO-N>:  <STOVO-N>;
                  <VENPASS>:  <SVEN>:
           $POS4 OR IF CORE2 HAS SNACBFLG AND SPAN1 ISNT NL THEN
                       CORE1 HAS NSENT1 OR NSENT2 OR NSENT3 OR (PRO
                       AND NOT NHUMAN) OR NULLNFLG OR FORTOVCFLG
                       OR TOVCFLG.
        HOUSED AT <SUBJECT>:  <OBES>:
           $POS4 OR IF CORE1 HAS SNACBFLG THEN CORE2 HAS NSENT1 OR
                       NSENT2 OR NSENT3 OR (PRO AND NOT NHUMAN) OR
                       NULLNFLG OR FORTOVCFLG OR TOVCFLG.
* DSN2:  AN SN STRING OR ASSERTION OCCURS AS THE RIGHT ADJUNCT OF AN
*       ADJECTIVE ONLY FOR CERTAIN
*       SUBCLASSES OF ADJECTIVES ("IT IS TRUE THAT HE CAME", *"IT IS
*       ROUND THAT HE CAME").
DSN2:   HOUSED AT <SN>, <ASSERTION>:  <RA>:
                    CORE HAS SNARAFLG.
           $PUNC OR INTER(P) % 1,1.
```

```
HOUSED AT <RA>; <LAR>;
   $POS4 OR IF CORE2 HAS SNARAFLG THEN CORE1 HAS ASENT1 OR
                ASENT3 OR VSENT1.
            IF CORE2 HAS SNARAFLG AND CORE1 HAS ASENT1 AND NOT
                ASENT3 THEN CORE HAS SUBJITFLG.
            CORE HAS SNARAFLG OF CORE2.
HOUSED AT <LAR>;  <ADJINRN>;
   $POS3 OR CORE2 HASNT SNARAFLG.
HOUSED AT <LAR>;  <NA>; <NULBEINGC>; <ASTG>;
          <NULBEINGO>;  <BEINGO>;
          <ASTG>;  <ADJN>; <CBJBE>; <PASSOBJ>; <NULOBES>;
                   <CBJECT>;
          <OBJBE>;  <OBJECTBE>; <OBJECT>; <ASOBJBE>; <PASSOBJ>;
                    <NULSASOBJBE>; <SOBJBE>;
          <ASCBJBE>;  <PASSOBJ>;
          <BEINGO>;  <CBJECTBE>;
          <NA>, <ADJN>;  <OBJECT>;
          <OBJECTBE>;  <OBJECT>;
          <OBJECT>;  <ASSERTIONVO>; <YESNOQVO>; <TSUBJVOVO>;
                     <VINGO>; <VENO>; <TOVO>; <VC>;
          <PASSOBJ>;  <VENPASS>;
          <VINGO>, <VENO>, <VO>, <VENPASS>, <TOVC>;  <OBJECT>;
                                                     <PASSOBJ>;
          <TOVO>;  <TOVO-N>;
          <VENPASS>;  <NULOBES>;
          <NULOBES>;  <OBES>;
            CORE HAS SNARAFLG, SUBJITFLG OF CORE2.
HOUSED AT <VINGSTG>;  <SUBJECT>;
            CORE HAS SUBJVINGFLG.
HOUSED AT <NWHSTG>;  <NSTG>;
            CORE HAS SUBJWHFLG.
HOUSED AT <NSTG>;  <SUBJECT>;
            CORE HAS SUBJWHFLG OF CORE2.
HOUSED AT <SN>;  <SUBJECT>;
            CORE HAS SUBJSNFLG.
HOUSED AT <SUBJECT>;  <ASSERTION>; <FORTOVO>; <NTOVC>; <SOBJBE>;
                      <SVEN>; <SASOBJBE>; <STOVC-N>; <YESNOQ>;
                      <TSUBJVO>;
            CORE HAS SUBJVINGFLG, SUBJSNFLG, SUBJWHFLG OF
                CORE2.
HOUSED AT <ASSERTIONVO>;  <ASSERTION>;
          <YESNOQVC>;  <YESNOQ>;
          <TSUBJVCVO>;  <TSUBJVO>;
          <TOVC>;  <FORTOVO>; <NTOVC>;
          <TOVO-N>;  <STOVO-N>;
          <VENPASS>;  <SVEN>;
          <OBJBE>;  <SOBJBE>;
          <NULSASCBJBE>;  <SASOBJBE>;
   $POS4 OR IF CORE2 HAS SNARAFLG THEN CORE1 HASNT SUBJSNFLG OR
                SUBJVINGFLG OR SUBJWHFLG OR NSENT1 OR NSENT2.
HOUSED AT <SUBJECT>;  <OBES>;
   $POS4 OR IF CORE1 HAS SNARAFLG THEN CORE2 HASNT SUBJSNFLG OR
                SUBJVINGFLG OR SUBJWHFLG OR NSENT1 OR NSENT2.
```

```
* DSN4:  A STRING OF SN CAN BE A POST-OBJECT ADJUNCT OF THE SUBJECT ONLY
*       IF THE SUBJECT IS A NOUN OF SUBCLASS NSENT1 OR NSENT2 AND THE
*       SUBJECT DOESN'T HAVE AN SN RIGHT ADJUNCT ("EVIDENCE IS PRESENTED
*       THAT GLUCAGON IS A PROTEIN").
DSN4:   HOUSED AT <SN>:  <RNSUBJ>; <RN>:
                  CORE HAS RNSNFLG.
        HOUSED AT <RN>:  <LNR>,1,2; <LNR>,2,2; <LPROR>;
                  <LNR>, <LPROR>:  <NSTG>;
                  <NSTG>:  <SUBJECT>;
                  <RNSUBJ>:  <REPSA>;
                  <REPSA>:  <SA>;
                  <SA>:  <YESNOQVO>,4; <TSUBJVOVO>,4; <ASSERTIONVO>,5;
                         <VENPASS>,6; <VINGO>,6; <TOVO>,7; <SOBJBE>,4;
                         <NULSASCBJBE>;
                  <NULSASCBJBE>:  <SASOBJBE>;
                  <SUBJECT>:  <ASSERTION>; <YESNOQ>; <TSUBJVO>;
                              <FORTOVO>; <NTOVO>; <STOVO-N>;
                              <SVEN>:
                    CORE HAS RNSNFLG OF CORE2.
        HOUSED AT <ASSERTIONVO>:  <ASSERTION>;
                  <YESNOQVO>:  <YESNOQ>;
                  <TSUBJVOVO>:  <TSUBJVO>;
                  <TOVO>:  <FORTOVO>; <NTOVO>;
                  <TOVO-N>:  <STOVO-N>;
                  <VENPASS>:  <SVEN>;
                  <OBJBE>:  <SOBJBE>;
                  <NULSASCBJBE>:  <SASOBJBE>:
           $POS4 OR IF CORE2 HAS RNSNFLG AND SPAN1 ISNT NL THEN
                       CORE1 HAS (NSENT1 OR NSENT2) AND NOT RNSNFLG.
* DSN5:  AN SN STRING IS ACCEPTABLE AS RIGHT ADJUNCT OF N ONLY IF N IS
*       NSENT1, NSENT2 OR NSENTP.  IF N IS NSENTP, N IS IN PN WITH
*       APPROPRIATE P AND THE ULTIMATE SUBJECT IS "IT".
DSN5:   HOUSED AT <RN>:  <LNR>1,2; <LNR>2,2:
           $POS4 OR IF CORE2 HAS RNSNFLG THEN CORE1 HAS NSENT1 OR
                       NSENT2 OR NSENTP.
                    IF CORE2 HAS RNSNFLG AND CORE1 HAS NSENTP THEN CORE
                       HAS NSENTPFLG.
        HOUSED AT <LNR>:  ALL OCCURRENCES EXCEPT <NSTG>;
                  <NSTG>:  ALL OCCURRENCES EXCEPT <NSTGO>:
                  <NSTGO>:  ALL OCCURRENCES EXCEPT <PN>:
           $POS4 OR CORE2 HASNT NSENTPFLG.
        HOUSED AT <LNR>:  <NSTG>;
                  <NSTG>:  <NSTGO>:
                    CORE HAS NSENTPFLG OF CORE2.
        HOUSED AT <NSTGO>:  <PN>:
                    IF CORE2 HAS NSENTPFLG THEN CORE HAS SUBJITFLG.
           $POS5 OR IF CORE2 HAS NSENTPFLG THEN CORE1 * NSENTP:PVAL
                       (CORE2) % NL.
        HOUSED AT <PN>:  <NULTSUBJVO>; <NULOBES>; <NPN>; <PNN>;
                         <PNTHATSVO>; <NULVINGSTGPN>; <OBJECT>;
                         <NULDPPN>; <OBJBE>; <PNSNWH>; <PASSOBJ>;
                  <NULTSUBJVO>:  <TSUBJVO>;
                  <NULOBES>:  <OBES>;
```

```
                    <NULVINGSTGPN>:  <VINGSTGPN>; <PNVINGSTG>;
                    <NULOPPN>:  <DP2PN>; <DP3PN>; <DP4PN>; <DP1PN>;
                    <OBJBE>:  <OBJECT>; <OBJECTBE>; <NULSASOBJBE>;
                              <ASOBJBE>; <PASSOBJ>;
                    <NPN>, <PNN>,<PNTHATSVO>, <VINGSTGPN>, <PNVINGSTG>,
                    <DP2PN>, <DP3PN>, <DP4PN>, <DP1PN>, <PNSNWH>:
                         <OBJECT>;
                    <ASOBJBE>, <DP1PN>, <PNSNWH>:  <PASSOBJ>:
                    <OBJECTBE>:  <OBJECT>;
                    <OBJECT>:  <ASSERTIONVO>; <YESNOQVO>; <TSUBJVOVO>;
                               <VINGO>; <VENO>; <TOVO>; <VC>;
                    <PASSOBJ>:  <VENPASS>;
                    <VINGO>, <VENO>, <VO>, <VENPASS>, <TOVC>:  <OBJECT>;
                                                               <PASSOBJ>;
                    <TOVC>:  <TOVO-N>;
                    <VENPASS>:  <NULOBES>;
                    <NULOBES>:  <OBES>:
                      CORE HAS SUBJITFLG OF CORE2.
          HOUSED AT <ASSERTIONVO>:  <ASSERTION>;
                    <YESNOQVO>:  <YESNOQ>;
                    <TSUBJVOVO>:  <TSUBJVO>;
                    <TOVC>:  <FORTOVO>; <NTOVO>;
                    <TOVC-N>:  <STOVO-N>;
                    <VENPASS>:  <SVEN>;
                    <OBJBE>:  <SOBJBE>;
                    <NULSASOBJBE>:  <SASOBJBE>:
             $POS4 OR IF CORE2 HAS SUBJITFLG THEN CORE1 HAS !IT!.
          HOUSED AT <SUBJECT>:  <OBES>; <TSUBJVO>:
                      IF CORE1 HAS SUBJITFLG THEN CORE2 HAS !IT!.
* DSN6:  IF THE TOVO-N OPTION OF RA IS TAKEN THEN THE HOST ADJECTIVE MUST
*        EITHER BE ADJ OF SUBCLASS ASENT1:AFORTO OR VING OF SUBCLASS
*        VSENT1. ("THE PROBLEM IS EASY TO SOLVE", "THAT IS ASTONISHING TO
*        HEAR").
DSN6:    HOUSED AT <TOVO-N>:  <RA>:
                     CORE HAS TOVO-NFLG.
         HOUSED AT <RA>:  <LAR>:
            $POS4 OR IF CORE2 HAS TOVO-NFLG THEN CORE1 HAS VSENT1 OR
                        ASENT1:AFORTO.
* DSN7:  IF SN IS IN RN THEN HOST NOUN IS NSENT1 OR NSENTP:ASHOULD.
*     IF SN IS IN RNSUBJ THEN THE SUBJECT IS NSENT1:ASHOULD.
DSN7:    HOUSED AT <C1SHOULD>:  <SN>:
                     CORE HAS C1SHOULDFLG.
         HOUSED AT <SN>:  <REPRN>; <RNSUBJ>;
                    <REPRN>:  <RN>;
                    <RNSUBJ>:  <REPSA>;
                    <REPSA>:  <SA>;
                    <SA>:  <ASSERTIONVO>; <YESNOQVO>; <TSUBJVOVO>;
                           <TOVO>; <NULSASOBJBE>; <VINGC>; <VENPASS>;
                    <TOVC>:  <TOVO-N>:
                      CORE HAS C1SHOULDFLG OF CORE2.
         HOUSED AT <RN>:  <LNR>,2,2:
                      CORE HASNT C1SHOULDFLG.
         HOUSED AT <RN>:  <LNR>,1,2:
```

```
      $POS4 OR IF CORE2 HAS C1SHOULDFLG THEN CORE1 HAS NSENT1:
                  ASHOULD OR NSENTP:ASHOULD.
      HOUSED AT <ASSERTIONVO>:  <ASSERTION>;
                <YESNOQVO>:  <YESNOQ>;
                <TSUBJVOVO>:  <TSUBJVO>;
                <VINGO>:  <SVINGO>;
                <TOVO>:  <FORTOVO>; <NTOVO>;
                <TOVO-N>:  <STOVO-N>;
                <VENPASS>:  <SVEN>;
                <OBJBE>:  <SOBJBE>;
                <NULSASCBJBE>:  <SASOBJBE>:
      $POS4 OR IF CORE2 HAS C1SHOULDFLG AND SPAN1 ISNT NL THEN
                  CORE1 HAS NSENT1:ASHOULD.
      HOUSED AT <OBJBE>:  <SOBJBE>:
                IF CORE HAS NSENT1:ASHOULD THEN CORE HAS
                  CKC1SHOULDFLG.
      HOUSED AT <SA>:  <OBJBE>,4:
      $POS4 OR IF CORE2 HAS C1SHOULDFLG THEN CORE1 HAS
                  OKC1SHOULDFLG.
* DSN8:  A STRING IN SN RARELY IF EVER OCCURS AS THE VALUE OF THE SUBJECT OF
*       A STRING IN SN OR RELATED STRINGS.
DSN8:  HOUSED AT <SN>:  <SUBJECT>:
                CORE HAS SUBJSNFLG.
       HOUSED AT <SUBJECT>:  <ASSERTION>;
                <ASSERTION>:  <WHETHS>; <WHS-N>; <PWHS>; <PWHS-PN>;
                              <WHNS-N>; <PWHNS>; <PWHNS-PN>:
                CORE HAS SUBJSNFLG OF CORE2.
       HOUSED AT <SUBJECT>:  <FORTOVO>; <NTOVO>; <SOBJBE>; <SVEN>;
                             <SASOBJBE>; <STOVO-N>:
                <ASSERTION>:  <S-N>; <THATS>; <C1SHOULD>; <SVO>;
                <WHETHS>, <WHS-N>, <PWHS>, <PWHS-PN>, <WHNS-N>,
                <PWHNS>, <PWHNS-PN>:  <SNWH>:
      $RARE OR CORE2 HASNT SUBJSNFLG.
* DSN9:  TOVC AS NON-SENTENTIAL RA ADJOINS ONLY ADJ:AASP OR
*        ASENT1:AFORTC ("HE IS APT TO DO IT", "HE IS NOBLE TO DO IT")
*        AND HAS A HUMAN SUBJECT.
DSN9:  HOUSED AT <TOVC>:  <RA>:
                CORE HAS RATOVOFLG.
       HOUSED AT <RA>:  <LAR>:
      $POS4 OR IF CORE2 HAS RATOVOFLG THEN CORE1 HAS AASP OR
                  ASENT1:AFORTO.
       HOUSED AT <LAR>:  <NA>; <NULBEINGO>; <ASTG>;
                <NULBEINGO>:  <BEINGO>;
                <ASTG>:  <ADJN>; <OBJBE>; <PASSOBJ>; <NULOBES>;
                         <OBJECT>;
                <OBJBE>:  <OBJECTBE>; <OBJECT>; <ASOBJBE>; <PASSOBJ>;
                          <NULSASOBJBE>; <SOBJBE>;
                <ASOBJBE>:  <PASSOBJ>;
                <BEINGO>:  <OBJECTBE>;
                <OBJECTBE>:  <OBJECT>;
                <OBJECT>:  <ASSERTIONVO>; <YESNOQVO>; <TSUBJVOVO>;
                           <VINGO>; <VENO>; <TOVO>; <VO>;
                <PASSOBJ>:  <VENPASS>;
```

```
                <VINGC>, <VENO>, <VO>, <VENPASS>, <TOVC>:  <OBJECT>;
                                                    <PASSOBJ>;
                <TOVC>:  <TOVC-N>;
                <VENPASS>:  <NULOBES>;
                <NULOBES>:  <OBES>:
                  CORE HAS RATOVOFLG OF CORE2.
        HOUSED AT <ASSERTIONVO>:  <ASSERTION>;
                <YESNOQVO>:  <YESNOQ>;
                <TSUBJVOVO>:  <TSUBJVO>;
                <TOVC>:  <FORTOVO>; <NTOVO>;
                <TOVO-N>:  <STOVO-N>;
                <VENPASS>:  <SVEN>;
                <OBJBE>:  <SOBJBE>;
                <NULSASOBJBE>:  <SASOBJBE>:
           $POS4 OR IF CORE2 HAS RATOVOFLG AND SPAN1 ISNT NL THEN
                        CORE1 HAS ((N AND (NHUMAN OR AGGREGATE)) OR
                        (PRO AND (NHUMAN OR PLURAL)),
* DSN10:  THE SPECIAL OBJECT STRING DPOFN OCCURS ONLY IN VINGOFN ("THE
*       WRITING DOWN OF THE RESULT"),
DSN10:  HOUSED AT <DP4>:  <OBJECT>:
                  CORE HAS DP4FLG,
        HOUSED AT <OBJECT>:  ALL OCCURRENCES EXCEPT <VINGOFN>:
           $POS3 OR CORE2 HASNT DP4FLG,
* DSN11:  TV IS NOT AN ELEMENT OF C1SHOULD OR SVO; WV IS NOT AN ELEMENT OF
*       SVO; V IS AN ELEMENT OF C1SHOULD OR SVO ("ASK THAT JUSTICE BE
*       DONE", "INSIST THAT HE SHOULD DO IT"),
DSN11:  HOUSED AT <LTVR>:  <VERB>:
                  CORE HAS TVFLG,
        HOUSED AT <LVR>:  <VERB>,2,3:
                  CORE HAS WVFLG,
        HOUSED AT <LVR>:  <VERB>,3,1:
                  CORE HAS VFLG,
        HOUSED AT <VERB>:  <ASSERTIONVO>;
                <ASSERTIONVO>:  <ASSERTION>:
                  CORE HAS TVFLG, WVFLG, VFLG OF CORE2,
        HOUSED AT <ASSERTION>:  <C1SHOULD>:
           $POS3 OR CORE2 HASNT TVFLG,
        HOUSED AT <ASSERTION>:  <SVO>:
           $POS3 OR CORE2 HASNT WVFLG OR TVFLG,
        HOUSED AT <ASSERTION>:  ALL OCCURRENCES EXCEPT <C1SHOULD>;
                                        <SVO>:
           $POS3 OR CORE2 HASNT VFLG,
*  WSN1:  CORRECT VERB FOR SN SUBJECT,
WSN1:   HOUSED AT <LVENR>:  <VENPASS>:
                  CORE HAS DVBVENPASSFLG,
        HOUSED AT <PASSOBJ>:  <VENPASS>:
                  IF CORE2 HAS DVBFLG AND NOT DVBVENPASSFLG THEN
                       CORE HASNT DVBVENPASSFLG,
        HOUSED AT <VENPASS>:  <NULOBES>; <OBJECTBE>;
                <VINGO>, <VENO>, <TOVO>, <VO>, <VENPASS>:  <OBJECT>;
                     <PASSOBJ>;
                <OBJECTBE>:  <OBJECT>;
                <TOVC>:  <TOVO-N>;
```

```
                <NULCBES>;  <OBES>;
                  CCRE HAS DVBVENPASSFLG OF CORE2,
        HOUSED AT <SUBJECT>;  <FORTOVO>; <NTOVO>; <SOBJBE>; <SVEN>;
                              <SASOBJBE>; <STOVO-N>; <YESNOQ>;
                              <TSUBJVO>;
                  CCRE HAS SUBJSNFLG OF CORE2,
        HOUSED AT <ASSERTIONVO>;  <ASSERTION>;
                  <YESNOQVO>;  <YESNOQ>;
                  <TSUBJVCVO>;  <TSUBJVO>;
                  <TOVC>;  <FORTCVO>; <NTOVO>;
                  <TOVO-N>;  <STOVO-N>;
                  <VENPASS>;  <SVEN>;
                  <OBJBE>;  <SOBJBE>;
                  <NULSASCBJBE>;  <SASOBJBE>;
           $POS4 OR IF CCRE1 HAS SUBJSNFLG THEN CORE2 HAS DOBFLG OR
                              VSENT1 OR VSENT2 OR VSENT3 OR VMOD OR VEXP,
           $POS4 OR IF CORE1 HAS SUBJSNFLG AND CORE2 HAS VSENT3 THEN
                              CORE2 HAS DVBVENPASSFLG,
        HOUSED AT <SUBJECT>;  <OBES>;
           $POS4 OR IF CORE2 HAS SUBJSNFLG THEN CORE1 HAS DOBFLG OR
                              VSENT1 OR VSENT2 OR VSENT3 OR VMOD OR VEXP,
           $POS4 OR IF CORE2 HAS SUBJSNFLG AND CORE1 HAS VSENT3 THEN
                              CORE1 HAS DVBVENPASSFLG,
*  WSN2:  CORRECT PREDICATE FOR SN SUBJECT,
WSN2:   HOUSED AT <ASTG>;  <CBJBE>;
                  CCRE HAS ASTGOBFLG,
        HOUSED AT <ASSERTIONVO>;  <ASSERTION>;
                  <YESNOQVO>;  <YESNOQ>;
                  <TSUBJVCVO>;  <TSUBJVO>;
                  <TOVC>;  <FORTOVO>; <NTOVC>;
                  <TOVO-N>;  <STOVO-N>;
                  <OBJBE>;  <SCBJBE>;
                  <NULSASCBJBE>;  <SASOBJBE>;
           $POS4 OR IF CORE1 HAS SUBJSNFLG AND CORE2 HAS DOBFLG AND
                              ASTGOBFLG AND (ADJ OR VING) THEN CORE2 HAS
                              ASENT1 OR ASENT2 OR VSENT1,
        HOUSED AT <SUBJECT>;  <OBES>;
           $POS4 OR IF CORE2 HAS SUBJSNFLG AND CORE1 HAS DOBFLG AND
                              ASTGOBFLG AND (ADJ OR VING) THEN CORE1 HAS
                              ASENT1 OR ASENT2 OR VSENT1,
*  WSN3:  WHEN THE OBJECT OF THE VERB "BE" IS AN SN STRING, IF THE SUBJECT
*   OF "BE" IS TOVO THEN THE OBJECT OF "BE" IS ALSO TOVO, AND IF THE
*   SUBJECT IS FORTOVC THEN THE OBJECT IS ALSO FORTOVC ("TO ASK A QUESTION
*   IS TO ANSWER IT"),
WSN3:   HOUSED AT <ASSERTIONVO>;  <ASSERTION>;
                  <YESNOQVO>;  <YESNOQ>;
                  <TSUBJVCVO>;  <TSUBJVO>;
                  <VINGO>;  <SVINGO>;
                  <TOVC>;  <FORTOVO>; <NTOVC>;
                  <TOVO-N>;  <STOVO-N>;
                  <VENPASS>;  <SVEN>;
                  <OBJBE>;  <SCBJBE>;
                  <NULSASCBJBE>;  <SASOBJBE>;
```

```
            $POS3 OR IF CORE1 HAS FORTOVOFLG AND CORE2 HAS DOBFLG
                          AND SNACBFLG THEN CORE2 HAS FORTOVOFLG,
            $POS3 OR IF CORE1 HAS TOVCFLG AND CORE2 HAS DOBFLG
                          AND SNACBFLG THEN CORE2 HAS TOVCFLG,
         HOUSED AT <SUBJECT>:  <OBES>:
            $POS3 OR IF CORE2 HAS FORTOVOFLG AND CORE1 HAS DOBFLG
                          AND SNACBFLG THEN CORE1 HAS FORTOVOFLG,
            $POS3 OR IF CORE2 HAS TOVCFLG AND CORE1 HAS DOBFLG
                          AND SNACBFLG THEN CORE1 HAS TOVCFLG,
*  WSN4:  IF VERB HAS LWR AND IS ELEMENT OF C1SHOULD THEN LWR IS
*   "SHOULD" ("INSIST THAT HE SHOULD DO IT" "INSIST HE SHOULD DO IT"),
WSN4:    HOUSED AT <LWR>:  <VERB>:
                          IF CORE2 HAS !SHOULD! THEN CORE HAS SHOULDFLG,
         HOUSED AT <VERB>:  <ASSERTIONVO>;
                   <ASSERTIONVO>:  <ASSERTION>:
                     CORE HAS SHOULDFLG OF CORE2,
         HOUSED AT <VERB>:  ALL OCCURRENCES EXCEPT <ASSERTIONVC>;
                   <ASSERTION>:  ALL OCCURRENCES EXCEPT <C1SHOULD>;
            $POS4 OR CORE2 HASNT SHOULDFLG,
         HOUSED AT <ASSERTION>:  <C1SHOULD>:
            $POS4 OR CORE2 HAS SHOULDFLG,
* WSN5:  C1SHOULD STRING WITHOUT "THAT" IS NOT TAKEN AS SUBJECT ("THAT
*   HE SHOULD LEAVE WAS TO BE EXPECTED" *"HE SHOULD LEAVE WAS TO BE
*   EXPECTED")
WSN5:    HOUSED AT !THAT!:  <C1SHOULD>:
                          CORE HAS THATFLG,
         HOUSED AT <C1SHOULD>:  <SN>:
                          CORE HAS THATFLG OF CORE2,
         HOUSED AT <SN>:  <SUBJECT>:
            $POS3 OR CORE2 HAS THATFLG,
*  WSN6:  IF WHETHS BEGINS WITH "IF" THEN IT IS NOT OCCURRING AS THE
*   SUBJECT,
WSN6:    HOUSED AT !IF!:  <NULWHETHS>:
                          CORE HAS IFFLG,
         HOUSED AT <NULWHETHS>:  <WHETHS>;
                   <WHETHS>:  <SNWH>;
                   <SNWH>:  <SN>:
                     CORE HAS IFFLG OF CORE2,
         HOUSED AT <SN>:  <SUBJECT>:
            $RARE OR CORE2 HASNT IFFLG.                                    *
* WSN7:  WHEN WHETHS OCCURS AS AN SA, IT CONTAINS AN "OR",  ("WHETHER OR
*     NOT YOU GO, I WILL GO", *"WHETHER YOU GO, I WILL GO.")
WSN7:    HOUSED AT <ORNOT>:  <WHETHS>,2;
                          CORE HAS ORNOTFLG,
         HOUSED AT <ORNOT>:  <WHETHS>,4:
            $POS3 OR CORE1 HASNT ORNOTFLG,
                          CORE HAS ORNOTFLG,
         HOUSED AT <WHETHS>:  <SAWH>:
            $POS3 OR CORE2 HAS ORNOTFLG,
* WSN8:  IN SAWHIICHSTG THE DEEPEST VERB IS SENTENCE OPERATOR VERB,
*    VSENT1 (SURPRISE, AMAZE) OR VSENT2 (PROVE, RESULT) OR VSENT3
*    (PREDICT, KNOW) OR EVENT VERB (HAPPEN, OCCUR), ("HE RECOVERED,
*    WHICH SURPRISED US"),  IF STRING HEAD IS "WHICH" DEEPEST VERB
```

```
*     IS VSENT1.
WSN8:    HOUSED AT <ASSERTION>:  <SAWHICHSTG>:
            $POS4 OR (CORE2 HAS DVBFLG AND (VSENT1 OR VSENT2 OR VSENT3
                        OR VEVENT) AND (IF CORE1 HAS WHICHFLG THEN
                        CORE2 HAS VSENT1)) OR (CORE2 HAS DOBFLG AND
                        (ADJ AND (ASENT1 OR ASENT3)) OR (VING AND
                        VSENT1)).
* WSN9:  SVINGO AS SUBJECT DOES NOT START WITH PRONOUN.
WSN9:    HOUSED AT <VINGO>:  <SVINGO>:
                     IF CORE1 HAS PRO THEN CORE HAS PROFLG.
         HOUSED AT <SVINGO>:  <SN>:
                     CORE HAS PROFLG OF CORE2.
         HOUSED AT <SN>:  <SUBJECT>:
            $POS3 OR CORE2 HASNT PROFLG.
* WSN10:  IN NSVINGO TPOS IS NOT A, AN, THE, SOME, EACH, EVERY.
*     (GIVING UP, HIS GIVING UP, NO GIVING UP, THIS GIVING UP.
*     *THE GIVING UP).
WSN10:   HOUSED AT <TPOS>:  <NSVINGO>:
            $POS4 OR CORE2 HASNT !A! OR !AN! OR !THE! OR TQUAN OR
                        EACHEVERY.
* WSN11:  LN2 IS NOT EMPTY UNLESS THE OF POSITION IS OCCUPIED OR VALUE
*     OF OBJECT IS DP4.  IF NO POST TPOS IN LN2, TPOS IS "A", "AN", "THE",
*     "SOME", "EACH", "EVERY".  ("HIS GIVING UP OF THE HOUSE".)
WSN11:   HOUSED AT <QPOS>:  <LN2>:
                     CORE HAS QPOSFLG.
         HOUSED AT <APOS>:  <LN2>:
                     CORE HAS APOSFLG.
         HOUSED AT <LN2>:  <VINGOFN>:
                     IF CORE2 HASNT QPOSFLG OR APOSFLG OR !A! OR !AN!
                        OR !THE! OR TQUAN OR EACHEVERY THEN CORE HAS
                        OFDP4FLG.
         HOUSED AT !OF!:  <VINGOFN>:
                     CORE HASNT OFDP4FLG.
                     CORE HAS OFFLG.
         HOUSED AT <NSTG>:  <OBJBE>;
                    <VINGSTG>:  <OBJECT>; <OBJECTBE>;
                    <NPN>, <NSTGO>:  <OBJECT>:
                     CORE HAS NSTARTFLG.
         HOUSED AT <ASTG>:  <OBJBE>:
                     IF CORE2 HAS VING THEN CORE HAS NSTARTFLG.
         HOUSED AT <OBJBE>:  <OBJECTBE>; <OBJECT>;
                    <OBJECTBE>:  <OBJECT>:
                     CORE HAS NSTARTFLG OF CORE2.
         HOUSED AT <OBJECT>:  <VINGOFN>:
                     IF CORE2 HAS DP4FLG THEN CORE HASNT OFDP4FLG.
                     IF CORE1 HAS OFFLG THEN CORE2 HAS NSTARTFLG.
         HOUSED AT <VINGOFN>:  ALL OCCURRENCES:
            $POS4 OR CORE2 HASNT OFDP4FLG.
```

```
*
*            VERB AND CENTER RESTRICTIONS
*
* DVC1:  OBJLIST RESTRICTIONS FOR NULL OBJECT.
DVC1:  HOUSED AT <VERB>:  <ASSERTIONVO>;
                  <VERB2>:  <TSUBJVOVO>; <YESNOQVO>;
                  <*VING>:  <VINGOFN>;
                  <LVENR>:  <VENO>; <VENPASS>;
                  <LVINGR>:  <VINGO>;
                  <LVR>:  <VO>; <TOVO>;
                  <VERB1>:  <YESNOQ>; <TSUBJVO>:
                    CORE HAS NULLOBJFLG.
       HOUSED AT <OBJECT>:  <ASSERTIONVO>; <TSUBJVOVO>; <YESNOQVO>;
                            <VINGOFN>; <VENO>; <VO>; <TOVO>; <VINGO>;
                  <PASSOBJ>:  <VENPASS>:
                    CORE HASNT NULLOBJFLG.
       HOUSED AT <RV>, <SA>, <ORNOT>:  <YESNOQVO>;
                  <RV>, <SA>:  <TSUBJVOVO>:
                    IF SPAN1 IS NL THEN CORE HAS NULLOBJFLG.
       HOUSED AT <YESNOQVO>:  <YESNOQ>;
                  <TSUBJVOVO>:  <TSUBJVO>:
                    IF CORE2 HAS NULLOBJFLG AND VERB2FLG THEN
                         NULLOBJFLG EL OBJLIST(CORE2).
                    IF CORE2 HAS NOT NULLOBJFLG OR (NULLOBJFLG AND
                         VERB2FLG) THEN CORE HASNT NULLOBJFLG.
       HOUSED AT <ASSERTIONVO>, <VINGOFN>, <VENO>, <VINGO>, <VO>,
                  <TOVO>:  ALL OCCURRENCES:
          $POS2 OR IF CORE2 HAS NULLOBJFLG THEN NULLOBJ EL OBJLIST
                         (CORE2).
       HOUSED AT <VENPASS>:  ALL OCCURRENCES:
          $POS2 OR IF CORE2 HAS NULLOBJFLG THEN NULLOBJ EL POBJLIST
                         (CORE2).
       HOUSED AT <VINGOFN>:  ALL OCCURRENCES:
          $POS2 OR IF CORE2 HAS NULLOBJFLG THEN CORE2 HASNT OFFLG.
*  DVC3:  IN YESNOQ AND TSUBJVO VERB2 HAS UNTENSED VERB V ONLY IF VERB1
*   HAS A W ("HAS HE ANSWERED THE PHONE"  *"HAS HE ANSWER THE PHONE"
*   "WILL HE ANSWER THE PHONE").
DVC3:  HOUSED AT <LWR>:  <VERB1>:
                    CORE HAS WFLG.
       HOUSED AT <VERB1>:  <YESNOQ>; <TSUBJVO>:
                    CORE HAS WFLG OF CORE2.
       HOUSED AT <VERB2>:  <YESNOQVO>; <TSUBJVOVO>:
                    CORE HAS VERB2FLG.
       HOUSED AT <YESNOQVO>:  <YESNOQ>;
                  <TSUBJVOVO>:  <TSUBJVO>:
          $POS3 OR IF CORE2 HAS VERB2FLG THEN CORE1 HAS WFLG.
          $POS3 OR IF CORE2 HASNT VERB2FLG THEN CORE1 HASNT WFLG.
*  WVC1:  AMONG INFLECTED VERBS ONLY "BE" AND "HAVE" MAY BE ADJOINED BY
*   "NOT".
WVC1:  HOUSED AT <RW>:  <LTVR>:
          $POS4 OR IF CORE2 HAS !NOT! THEN CORE1 HAS VBE OR VHAVE.
*  WVC2:  UNLIKE OTHER VERBS THE INFINITIVE "BE" IS NOT PRECEDED BY
*   BY "DO" ("THEY DON'T LIVE HERE" *"THEY DON'T BE HERE").
```

```
WVC2:  HOUSED AT <LVR>;  <VERB>,2,3;
          $PCS4 OR IF CORE2 HAS VBE THEN CORE1 HASNT VDO.
*  WVC3:  VENO IS NOT AN ACCEPTABLE VALUE OF OBJECT IN VENO NOR IN VINGO
*   AFTER "BE" (*"HE HAS HAD HAD GOOD LUCK").
WVC3:  HOUSED AT <VENC>;  <OBJECT>;
                  CORE HAS VENOFLG,
       HOUSED AT <OBJECT>;  <VENO>;
          $PCS3 OR CORE2 HASNT VENOFLG.
       HOUSED AT <OBJECT>;  <VINGO>;
                  CORE HAS VENOFLG OF CORE2,
       HOUSED AT <VINGO>;  <OBJECTBE>;
          $PCS3 OR CORE2 HASNT VENOFLG.
*  WVC4:  THE CORE OF LVENR IS NOT VNOPASS (*"HE WAS EXISTED").
WVC4:  HOUSED AT <LVENR>;  <VENPASS>;
          $PCS4 OR CORE2 HAS POBJLIST.
* WVC5:  IN YESNOQ AND TSUBJVO, ONLY "BE", "HAVE" OR W CAN OCCUR IN
*    VERB1 POSITION.  ("IS HE WORKING ON THE BOOK?"  *"WORKS HE ON
*    THE BOOK?").
WVC5:  HOUSED AT <LTVR>;  <VERB1>;
          $PCS4 OR CORE2 HAS VHAVE OR VBE.
*  WVC6:  IN OBES ULTIMATE VERB IS VBE OR BE REPLACER.
WVC6:  HOUSED AT <VERB3>;  <OBES>;
          $PCS4 OR CORE2 HAS VBE OR VHAVE OR BEREP.
       HOUSED AT <VERB4>;  <OBES>;
          $PCS4 OR IF CORE1 HAS VHAVE THEN CORE2 HAS VBE.
* WVC8:  PN AS THE OBJECT OF BE CANNOT HAVE P="SINCE" UNLESS THE ULTIMATE
*    SUBJECT IS "IT".  ("IT IS SINCE TUESDAY THAT THEY HAVE BEEN VISITING"
*    *"THEIR VISIT IS SINCE TUESDAY").
WVC8:  HOUSED AT <ASSERTIONVC>;  <ASSERTION>;
                 <YESNOQVC>;  <YESNOQ>;
                 <TSUBJVOVO>;  <TSUBJVO>;
                 <TOVC>;  <FORTOVO>;
          $PCS4 OR IF CORE2 HAS DOBFLG AND !SINCE! THEN CORE1 HAS !IT!.
       HOUSED AT <PN>;  <NULOBES>;
          $PCS4 OR CORE2 HASNT !SINCE!.
```

```
*
*                    WH-STRING RESTRICTIONS
*
*  WWH1A:  PASS RELEVANT INFORMATION UP TO ULTIMATE OBJECT.
WWH1A:  HOUSED AT <VENO>, <VO>, <VINGO>, <PVINGO>, <NPVINGO>, <SVEN>,
                  <OBJECTBE>, <TOVO>, <NTOVO>, <FORTOVO>, <ASSERTION>,
                  <WHETHS>, <SNWH>, <NPSNWH>, <NTHATS>, <THATS>,
                  <C1SHOULD>, <PNTHATSVO>:  <OBJECT>;
                  <VO>, <VINGO>, <PVINGO>, <VENPASS>, <TOVO>, <SNWH>,
                  <FORTOVO>, <THATS>:  <PASSOBJ>;
                  <OBJECT>:  <YESNOQVO>; <ASSERTIONVO>; <VENO>; <VO>;
                             <VINGO>; <TOVO>;
                  <YESNOQVO>:  <YESNOQ>;
                  <ASSERTIONVO>:  <ASSERTION>;
                  <VINGO>:  <OBJECTBE>; <PVINGO>;
                  <PVINGO>:  <NPVINGO>;
                  <PASSOBJ>:  <VENPASS>;
                  <VENPASS>:  <OBJECTBE>; <SVEN>;
                  <TOVO>:  <OBJECTBE>; <NTOVO>; <FORTOVO>;
                  <FORTOVO>:  <SN>;
                  <SN>:  <SUBJECT>; <OBJECTBE>;
                  <ASSERTION>:  <NULNTHATS>; <WHETHS>; <THATS>;
                                <C1SHOULD>;
                  <WHETHS>:  <SNWH>;
                  <SNWH>:  <SN>; <NPSNWH>;
                  <THATS>:  <SN>; <NULNTHATS>;
                  <C1SHOULD>:  <SN>; <PNTHATSVO>;
                  <SUBJECT>:  <ASSERTION>;
                  <NUL2WHQ-N>:  <WHQ-N>; <WHNQ-N>:
                    CORE HAS SOMITFLG, OOMITFLG, POMITFLG OF CORE2.
        HOUSED AT <OBJECT>:  ALL OCCURRENCES EXCEPT <YESNOQVO>; <VENO>;
                             <ASSERTIONVO>; <VINGO>; <VO>; <TOVO>;
                  <VINGO>:  ALL OCCURRENCES EXCEPT <OBJECT>; <PASSOBJ>;
                            <OBJECTBE>; <PVINGO>;
                  <PVINGO>:  ALL OCCURRENCES EXCEPT <OBJECT>;
                             <PASSOBJ>; <NPVINGO>;
                  <VENPASS>:  ALL OCCURRENCES EXCEPT <OBJECTBE>;
                              <SVEN>; <PASSOBJ>;
                  <TOVO>:  ALL OCCURRENCES EXCEPT <OBJECT>; <PASSOBJ>;
                           <OBJECTBE>; <SN>; <FORTOVO>; <NTOVO>; <RA>;
                           <TOVO-N>;
                  <SN>:  ALL OCCURRENCES EXCEPT <SUBJECT>; <OBJECTBE>;
                         <REPRN>; <RA>;
                  <ASSERTION>:  ALL OCCURRENCES EXCEPT <NULNTHATS>;
                                <WHETHS>; <THATS>; <C1SHOULD>; <OBJECT>;
                                <NUL2WHQ-N>; <PWHS-PN>; <PWHNS-PN>;
                                <WHS-N>; <WHNS-N>; <WHATS-N>; <THATS-N>;
                                <S-N>; <WHEVERS-N>;
                  <FORTOVO>:  ALL OCCURRENCES EXCEPT <OBJECT>; <SN>;
                              <PASOBJ>; <FORTOVO-N>;
                  <WHETHS>:  ALL OCCURRENCES EXCEPT <OBJECT>; <SNWH>;
                  <SNWH>:  ALL OCCURRENCES EXCEPT <PASSOBJ>; <OBJECT>;
                           <SN>; <NPSNWH>;
```

```
          <SUBJECT>:  ALL OCCURRENCES EXCEPT <ASSERTION>;
          <YESNOQ>:  ALL OCCURRENCES EXCEPT <NUL2WHQ-N>;
                     <PWHQ-PN>; <PWHNQ-PN>:
   $POS2 OR CORE HASNT SOMITFLG OR OOMITFLG OR POMITFLG,
HOUSED AT <TOVC>:  <RA>; <SN>:
            IF CORE2 HAS SOMITFLG THEN CORE HAS SOMITR1FLG,
            IF CORE2 HAS OOMITFLG OR NULLWHNOFLG THEN CORE HAS
                 COMITR1FLG,
            IF CORE2 HAS POMITFLG THEN CORE HAS POMITR1FLG,
HOUSED AT <SN>:  <RA>; <REPRN>;
          <REPRN>:  <RN>;
          <RN>:  <LNR>,1,2; <LNR>,2,2; <LPROR>;
          <LNR>, <LPROR>:  <NSTG>;
          <NSTG>:  <NSTGO>;
          <NSTGO>:  <PN>;
          <PN>:  <OBJBE>; <RNP>:
            CORE HAS SOMITR1FLG, OOMITR1FLG, POMITR1FLG OF
                 CORE2,
HOUSED AT <LNR>:  ALL OCCURRENCES EXCEPT <NSTG>;
          <NSTG>:  ALL OCCURRENCES EXCEPT <NSTGO>;
          <NSTGO>:  ALL OCCURRENCES EXCEPT <PN>;
          <PN>:  ALL OCCURRENCES EXCEPT <OBJBE>; <RNP>:
   $POS2 OR CORE2 HASNT SOMITR1FLG OR OOMITR1FLG OR POMITR1FLG,
HOUSED AT <RA>:  <LAR>;
          <OBJBE>:  <NULSASOBJBE>; <SOBJBE>;
          <RNP>:  <REPRN>:
            IF CORE2 HAS SOMITR1FLG THEN CORE HAS SOMITR2FLG,
            IF CORE2 HAS OOMITR1FLG THEN CORE HAS OOMITR2FLG,
            IF CORE2 HAS POMITR1FLG THEN CORE HAS POMITR2FLG,
HOUSED AT <REPRN>:  <RN>;
          <RN>:  <LNR>,1,2; <LNR>2,2; <LPROR>;
          <LNR>, <LPROR>:  <NSTG>;
          <NSTG>:  <OBJBE>; <NSTGO>;
          <LAR>:  <ASTG>;
          <ASTG>:  <OBJBE>;
          <OBJBE>:  <NULSASOBJBE>; <SOBJBE>;
          <NULSASOBJBE>:  <SASOBJBE>;
          <NSTGO>:  <PN>; <NPVINGSTG>; <PNN>; <NPN>; <NTHATS>;
                    <NPSVINGO>; <NPVINGO>; <NPSNWH>; <ND>;
                    <NPVINGSTG>; <ADJN>; <NA>; <NN>,1; <NN>,3;
          <PN>:  <OBJBE>; <PSTG>; <NPVINGSTG>; <PNN>; <NPN>:
            CORE HAS SOMITR2FLG, OOMITR2FLG, POMITR2FLG OF
                 CORE2,
HOUSED AT <LNR>:  ALL OCCURRENCES EXCEPT <NSTG>;
          <NSTG>:  ALL OCCURRENCES EXCEPT <NSTGO>; <SUBJECT>;
                   <OBJBE>;
          <NSTGO>:  ALL OCCURRENCES EXCEPT <PN>; <OBJECT>;
                    <PASSOBJ>; <NPVINGSTG>; <PNN>; <NPN>;
                    <NPSVINGO>; <NPVINGO>; <NPSNWH>; <NTHATS>;
                    <NPVINGSTG>; <ND>; <ADJN>; <NA>; <NN>,1;
                    <NN>,3;
          <PN>:  ALL OCCURRENCES EXCEPT <OBJECT>; <PASSOBJ>;
                 <PSTG>; <OBJBE>; <PNTHATSVO>; <NPVINGSTG>;
```

```
                    <PNN>; <NPN>;
                <SOBJBE>, <PSTG>:  ALL OCCURRENCES EXCEPT <OBJECT>;
                <ASTG>:  ALL OCCURRENCES EXCEPT <OBJBE>; <OBJECT>;
                    <PASSOBJ>:
          $POS2 OR CORE2 HASNT SOMITR2FLG OR OOMITR2FLG OR POMITR2FLG.
       HOUSED AT <NSTG>:  <SUBJECT>;
                <PN>:  <PNTHATSVO>;
                <OBJBE>:  <OBJECTBE>;
                <PN>, <NPVINGSTG>, <PNN>, <NPN>, <NTHATS>, <SOBJBE>,
                <NPSVINGO>, <NPVINGO>, <NPSNWH>, <ND>, <SASOBJBE>,
                <NPVINGSTG>, <ADJN>, <NA>, <NN>, <ASTG>, <NSTGO>,
                <OBJBE>, <PSTG>:  <OBJECT>;
                <ASTG>, <OBJBE>, <NSTGO>, <PN>:  <PASSOBJ>:
                  IF CORE2 HAS SOMITR2FLG THEN CORE HAS SOMITFLG.
                  IF CORE2 HAS OOMITR2FLG THEN CORE HAS OOMITFLG.
                  IF CORE2 HAS POMITR2FLG THEN CORE HAS POMITFLG.
       HOUSED AT <OBJBE>:  <OBJECT>; <OBJECTBE>; <PASSOBJ>:
                  IF CORE2 HAS SOMITR1FLG THEN CORE HAS SOMITFLG.
                  IF CORE2 HAS OOMITR1FLG THEN CORE HAS OOMITFLG.
                  IF CORE2 HAS POMITR1FLG THEN CORE HAS POMITFLG.
*  WWH1B:  CHECK THAT PROPER ELEMENT IS OMITTED FOR GIVEN STRING.
WWH1B:  HOUSED AT <YESNOQ>:  <NUL2WHQ=N>;
                <FORTOVC>:  <FORTOVO=N>:
          $POS2 OR CORE2 HAS SOMITFLG OR OOMITFLG.
       HOUSED AT <ASSERTION>:  <WHS=N>; <WHNS=N>; <WHATS=N>; <S=N>;
                    <THATS=N>; <WHEVERS=N>; <SAWHICHSTG>;
                <FORTOVC>:  <FORTOVO=N>:
          $POS2 OR CORE2 HAS SOMITFLG OR OOMITFLG OR NULLSUBJFLG.
       HOUSED AT <TOVC>:  <TOVO=N>:
          $POS2 OR CORE2 HAS SOMITFLG OR OOMITFLG OR NULWHNOFLG OR
                    NULLWHPOFLG.
       HOUSED AT <YESNOQ>:  <PWHQ=PN>; <PWHNQ=PN>;
                <ASSERTION>:  <PWHS=PN>; <PWHNS=PN>:
          $POS2 OR CORE2 HAS POMITFLG.
*  WWH1C:  NO MORE THAN ONE OMITTED ELEMENT PER STRING.
WWH1C:  HOUSED AT <ASSERTIONVO>:  <ASSERTION>;
                <OBJBE>:  <SOBJBE>;
                <NULSASCBJBE>:  <SASOBJBE>;
                <TOVC>:  <NTOVC>;
                <RV>:  <ASSERTIONVC>; <YESNOQ>; <VENO>; <VENPASS>;
                    <VINGC>; <VO>; <TOVO>;
                <C1SHOULD>:  <PNTHATSVO>;
                <VENPASS>:  <SVEN>;
                <PVINGO>:  <NPVINGC>;
                <SNWH>:  <NPSNWH>;
                <NULNTHATS>:  <NTHATS>;
          $POS2 OR IF CORE2 HAS NULLSUBJFLG OR NULLWHNCFLG OR
                  NULLWHPOFLG OR POSTOBJFLG OR ADJOMITFLG OR
                  NULLOBJBEFLG OR SOMITFLG OR OOMITFLG OR POMITFLG
                  OR SOMITR1FLG OR OOMITR1FLG OR POMITR1FLG OR
                  SOMITR2FLG OR OOMITR2FLG OR POMITR2FLG
                  THEN CORE1 HASNT NULLSUBJFLG OR NULLWHNOFLG OR
                  NULLWHPOFLG OR POSTOBJFLG OR ADJOMITFLG OR
```

```
                    NULLOBJBEFLG OR SOMITFLG OR OOMITFLG OR POMITFLG
                    OR SOMITR1FLG OR OOMITR1FLG OR POMITR1FLG OR
                    SOMITR2FLG OR OOMITR2FLG OR POMITR2FLG
          $POS2 OR IF CORE1 HAS NULLSUBJFLG OR NULLWHNOFLG OR
                    NULLWHPOFLG OR POSTOBJFLG OR ADJOMITFLG OR
                    NULLOBJBEFLG OR SOMITFLG OR OOMITFLG OR POMITFLG
                    OR SOMITR1FLG OR OOMITR1FLG OR POMITR1FLG OR
                    SOMITR2FLG OR OOMITR2FLG OR POMITR2FLG
                    THEN CORE2 HASNT NULLSUBJFLG OR NULLWHNOFLG OR
                    NULLWHPOFLG OR POSTOBJFLG OR ADJOMITFLG OR
                    NULLOBJBEFLG OR SOMITFLG OR OOMITFLG OR POMITFLG
                    OR SOMITR1FLG OR OOMITR1FLG OR POMITR1FLG OR
                    SOMITR2FLG OR OOMITR2FLG OR POMITR2FLG
*  WHH2:  DETECT A ZEROED SUBJECT AND SET UP FOR TESTING CONDITIONS FOR ITS
*  ACCEPTABILITY:  "A BOOK WHICH IS INTERESTING"  "A BOOK THEY SAY IS
*  INTERESTING",
WHH2:   HOUSED AT <SA>:  <SASOBJBE>,2; <SVEN>,2; <SOBJBE>,2;
                         <SOBJBE>,4; <NTOVO>,3; <ASSERTION>,3;
                  IAS!:  <SASOBJBE>;
                  <VENPASS>:  <SVEN>;
                  <OBJBE>:  <SOBJBE>;
                  <ASSERTIONVO>:  <ASSERTION>;
                  <RV>, <TOVC>:  <NTOVO>;
                    IF SPAN1 IS NL THEN CORE HAS NULLSUBJFLG.
        HOUSED AT <SA>:  <ASSERTION>,1:
                    CORE HAS NULLSUBJFLG.
        HOUSED AT <SUBJECT>:  <ASSERTION>:
                    CORE HASNT NULLSUBJFLG.
        HOUSED AT <SVEN>, <SOBJBE>:  ALL OCCURRENCES EXCEPT <OBJECT>:
                  <ASSERTION>:  ALL OCCURRENCES EXCEPT <C1SHOULD>;
                                <NULNTHATS>; <SVO>; <OBJECT>; <WHS-N>;
                                <NUL2WHQ-N>; <WHNS-N>; <WHATS-N>;
                                <THATS-N>; <S-N>; <WHEVERS-N>;
                                <SAWHICHSTG>:
           $POS2 OR CORE2 HASNT NULLSUBJFLG.
        HOUSED AT <ASSERTION>:  <C1SHOULD>:
           $POS2 OR IF CORE2 HAS NULLSUBJFLG THEN CORE1 HASNT THATFLG.
        HOUSED AT <ASSERTION>:  <C1SHOULD>; <NULNTHATS>; <SVO>;
                  <NULNTHATS>:  <NTHATS>:
                    CORE HAS NULLSUBJFLG OF CORE2.
        HOUSED AT <C1SHOULD>:  <SN>; <PNTHATSVO>;
                  <C1SHOULD>, <NTHATS>, <SVO>, <ASSERTION>, <SASOBJBE>,
                  <SOBJBE>, <NTOVC>, <SVEN>:  <OBJECT>;
                    IF CORE2 HAS NULLSUBJFLG THEN CORE HAS SOMITFLG.
*  WHH3A:  DETECT ZERCED CBJECT IN WH-STRING AND SET UP FOR TESTING
*  CONDITIONS FOR ITS ACCEPTABILITY:  "A BOOK I THINK I READ".
WHH3A:  HOUSED AT <NSTGO>:  <NN>,1;
                  <*P>:  <PN>; <PNTHATS>;
                  <PN>, <SA>:  <PNN>:
                  <*DP>:  <DP2>; <DP4>;
                  <ASTG>:  <ADJN>:
                    CORE HAS NULLWHNOFLG.
```

```
HOUSED AT <VERB>:  <ASSERTIONVO>;
          <VERB2>:  <YESNOQVO>;
          <LVENR>:  <VENO>;
          <LVINGR>:  <VINGO>;
          <LVR>:  <VO>; <TOVO>:
            IF NSTGO EL OBJLIST(CORE2) THEN CORE HAS
                 NULLWHNOFLG.
HOUSED AT <LVENR>:  <VENPASS>:
            IF NSTGO EL POBJLIST(CORE2) THEN CORE HAS
                 NULLWHNOFLG.
HOUSED AT <NSTGO>:  <NN>,3; <PNN>; <PNTHATS>; <ADJN>; <PN>;
          <NULNV>:  <DP2>; <DP4>
          <OBJECT>:  <ASSERTIONVO>; <YESNOQVO>; <VENO>; <VO>;
                     <VINGO>; <TOVO>;
          <PASSOBJ>:  <VENPASS>:
            CORE HASNT NULLWHNOFLG.
HOUSED AT <SA>:  <NPN>;
          <SA>, <PVINGSTG>:  <NPVINGSTG>;
          <SA>, <PSVINGO>:  <NPSVINGO>;
          <SA>, <PVINGO>:  <NPVINGO>;
          <SA>, <*P>:  <NPSNWH>;
          <*DP>:  <DP3>;
          <RV>, <SA>, <NULNTHATS>:  <NTHATS>;
          <RV>, <SA>, <SNWH>:  <NSNWH>;
          <SA>, <LAR>:  <NA>;
          <SA>, <DSTG>:  <ND>:
            IF SPAN1 IS NL THEN CORE HAS NULWHNOFLG.
 HOUSED AT <VENPASS>:  ALL OCCURRENCES EXCEPT <OBJECTBE>;
                       <SVEN>; <PASSOBJ>;
          <PN>:  ALL OCCURRENCES EXCEPT <OBJECT>; <PASSOBJ>;
                 <RN>; <RV>;
          <DP2>:  ALL OCCURRENCES EXCEPT <OBJECT>;
          <DP3>:  ALL OCCURRENCES EXCEPT <OBJECT>;
          <DP4>:  ALL OCCURRENCES EXCEPT <OBJECT>;
          <VINGO>:  ALL OCCURRENCES EXCEPT <OBJECT>; <PVINGO>;
                    <PASSOBJ>; <OBJECTBE>;
          <TOVO>:  ALL OCCURRENCES EXCEPT <OBJECT>; <PASSOBJ>;  *
                   <FORTOVO>; <NTOVO>; <OBJECTBE>; <TOVO-N>;
                   <SN>; <RA>:
    $POS2 OR CORE2 HASNT NULLWHNOFLG.
 HOUSED AT <NN>, <PNTHATS>, <PNN>, <ADJN>, <NPN>, <NPVINGSTG>,
          <NPVINGO>, <NPSVINGO>, <NPSNWH>, <NTHATS>, <NSNWH>,
          <NA>, <ND>, <DP2>, <DP3>, <DP4>, <PN>,<VENO>, <VO>,
          <VINGO>, <TOVO>:  <OBJECT>;
          <PN>, <VO>, <VINGO>, <TOVO>:  <PASSOBJ>;
          <ASSERTIONVO>:  <ASSERTION>;                          *
          <YESNOQVO>:  <YESNOQ>;
          <VINGO>:  <OBJECTBE>; <PVINGO>;
          <TOVO>:  <OBJECTBE>; <FORTOVO>; <NTOVO>;
          <VENPASS>:  <OBJECTBE>; <PASSOBJ>; <SVEN>:
            IF CORE2 HAS NULLWHNOFLG THEN CORE HAS OOMITFLG.
*  WWH3B:  DETECT ZERCED CBJECT CF PREPOSITION IN ADJUNCT POSITION AND SET
*  UP FOR TESTING CONDITICNS FOR ITS ACCEPTABILITY:  "THE PAPER WE NEED A
```

```
*  COPY OF",  "THE PAPER I HAVE A COPY OF A COPY OF",
WWH3B:  HOUSED AT <PN>:  <REPRV>:
                  IF CORE2 HAS NULLWHNOFLG THEN CORE HAS POSTOBJFLG.
        HOUSED AT <REPRV>:  <RV>:
                  CORE HAS POSTOBJFLG OF CORE2.
        HOUSED AT <RV>:  ALL OCCURRENCES EXCEPT <ASSERTIONVO>; <VENO>;
                         <YESNOQVO>; <VENPASS>; <VINGO>; <VO>; <TOVO>:
           $POS2 OR CORE2 HASNT POSTOBJFLG.
        HOUSED AT <PN>:  <RNP>:
                  IF CORE2 HAS NULLWHNOFLG THEN CORE2 HAS !OF! OR
                       !FOR!.
                  IF CORE2 HAS NULLWHNOFLG THEN CORE HAS ADJOMITFLG.
        HOUSED AT <RNP>:  <REPRN>;
                  <REPRN>:  <RN>;
                  <RN>:  <LNR>,1,2;
                  <LNR>:  <NSTG>;
                  <NSTG>:  <NSTGO>;
                  <NSTGO>:  <PN>;
                  <PN>:  <RN>:
                    CORE HAS ADJOMITFLG OF CORE2.
        HOUSED AT <RN>:  <LNR>,1,2:
           $POS2 OR IF CORE2 HAS ADJOMITFLG THEN CORE1 HAS N OR VING.
        HOUSED AT <RN>:  ALL OCCURRENCES EXCEPT <LNR>,1,2;
                  <LNR>:  ALL OCCURRENCES EXCEPT <NSTG>;
                  <NSTG>:  ALL OCCURRENCES EXCEPT <NSTGO>;
                  <NSTGO>:  ALL OCCURRENCES EXCEPT <PN>; <OBJECT>;
                            <PASSOBJ>;
                  <PN>:  ALL OCCURRENCES EXCEPT <RN>:
           $POS2 OR CORE2 HASNT ADJOMITFLG.
        HOUSED AT <RV>:  <ASSERTIONVO>; <YESNOQVO>; <VENO>; <VENPASS>;
                         <VINGO>; <VO>; <TOVO>:
                    IF CORE2 HAS POSTOBJFLG THEN CORE HAS OOMITFLG.
        HOUSED AT <NSTGO>:  <OBJECT>; <PASSOBJ>:
                    IF CORE2 HAS ADJOMITFLG THEN CORE HAS OOMITFLG.
*  WWH4:  DETECT ZEROED OBJECT-OF-BE AND SET UP FOR TESTING FOR ITS
*  ACCEPTABILITY:  "WHAT SHE IS IS PRETTY."  "NEARBY IS WHERE SHE IS."
WWH4:   HOUSED AT <VERB>:  <ASSERTIONVO>:
                    IF CORE2 HAS VBE OR BEREP THEN CORE HAS
                         NULLOBJBEFLG.
        HOUSED AT <SUBJECT>:  <SOBJBE>;
                  <SA>:  <SOBJBE>,2;
                  !AS!:  <SASOBJBE>; <ASOBJBE>;
                    CORE HAS NULLOBJBEFLG.
        HOUSED AT <SA>:  <SOBJBE>,4:
                    IF SPAN1 IS NL THEN CORE HAS NULLOBJFLG.
        HOUSED AT <OBJECT>:  <ASSERTIONVO>;
                  <OBJBE>:  <SOBJBE>; <ASOBJBE>;
                  <NULSASOBJBE>:  <SASOBJBE>:
                    CORE HASNT NULLOBJBEFLG.
        HOUSED AT <ASSERTIONVO>:  <ASSERTION>;
                  <SOBJBE>, <SASOBJBE>, <ASOBJBE>:  <OBJECT>;
                  <ASOBJBE>:  <PASSOBJ>:
                    IF CORE2 HAS NULLOBJBEFLG THEN CORE HAS SOMITFLG.
```

```
        HOUSED AT <SOBJBE>:  ALL OCCURRENCES EXCEPT <OBJECT>:
           $POS2 OR CORE2 HASNT NULLOBJBEFLG.
*  WWH5:  DETECT ZEROED PREPOSITIONAL PHRASE OCCURRING AS OBJECT AND SET
*  UP FOR TESTING CONDITIONS FOR ITS ACCEPTABILITY:  "A MAN ON WHOM YOU
*  CAN RELY".
WWH5:   HOUSED AT <VERB>:  <ASSERTIONVO>:
                  <VERB2>:  <YESNOQVO>;
                  <LVENR>:  <VENO>;
                  <LVINGR>:  <VINGO>;
                  <LVR>:  <VO>; <TOVO>:
                    IF PN EL OBJLIST(CORE2) THEN CORE HAS
                         NULLWHPOFLG.
        HOUSED AT <LVENR>:  <VENPASS>:
                    IF PN EL POBJLIST(CORE2) THEN CORE HAS
                         NULLWHPOFLG.
        HOUSED AT <OBJECT>:  <ASSERTIONVO>; <YESNOQVO>; <VENO>; <VO>;
                             <VINGO>; <TOVO>;
                  <PASSOBJ>:  <VENPASS>:
                    CORE HASNT NULLWHPOFLG.
        HOUSED AT <NSTGO>:  <NPN>:
                    CORE HAS NULLWHPOFLG.
        HOUSED AT <PN>:  <NPN>:
                    IF SPAN1 IS NL THEN CORE HAS NULLWHNOFLG ELSE CORE
                         HASNT NULLWHPOFLG.
        HOUSED AT <VENO>, <VO>, <VINGO>, <TOVO>:  <OBJECT>;
                  <PN>, <VO>, <VINGO>, <TOVO>:  <PASSOBJ>;
                  <ASSERTIONVO>:  <ASSERTION>;
                  <YESNOQVO>:  <YESNOQ>;
                  <VINGO>:  <OBJECTBE>; <PVINGO>;
                  <TOVO>:  <OBJECTBE>; <FORTOVO>; <NTOVO>;
                  <VENPASS>:  <OBJECTBE>; <PASSOBJ>; <SVEN>:
                    IF CORE2 HAS NULLWHPOFLG THEN CORE HAS POMITFLG.
*  WWH6A:  IF SUBJECT OR OBJBE OMITTED THEN WH-STRING DOES NOT BEGIN WITH
*  "WHOM" OR "WHOMEVER".  IF OBJECT OR OBJECT OF PREPOSITION OMITTED THEN
*  WH-STRING DOES NOT BEGIN WITH "WHO" OR "WHOEVER".
WWH6A:  HOUSED AT !WHOM!:  <NULWHS-N>; <NUL1WHQ-N>;
                  !WHOMEVER!:  <NULWHEVERS-N>:
                    CORE HAS ACCFLG.
        HOUSED AT !WHO!:  <NULWHS-N>; <NUL1WHQ-N>;
                  !WHOEVER!:  <NULWHEVERS-N>:
                    CORE HAS NOMFLG.
        HOUSED AT <NULWHS-N>:  <WHS-N>;
                  <NULWHQ-N>:  <WHQ-N>;
                  <NULWHEVERS-N>:  <WHEVERS-N>;
                    CORE HAS NOMFLG, ACCFLG OF CORE2.
        HOUSED AT <NUL2WHQ-N>:  <WHQ-N>:
                  <ASSERTION>:  <WHS-N>; <WHEVERS-N>:
           $SS  OR  IF CORE2 HAS SOMITFLG THEN CORE1 HASNT ACCFLG.
           $WS  OR  IF CORE2 HAS OOMITFLG THEN CORE1 HASNT NOMFLG.
*  WWH6B:  "WHO" AND "WHOM" TAKE HUMAN HOSTS.  "WHICH" TAKES NONHUMAN HOST.
WWH6B:  HOUSED AT !WHO!, !WHOM!:  <NULWHS-N>;
                  !WHOM!:  <NULPWHS>; <NULPWHS-PN>;
                    CORE HAS HUMFLG.
```

```
        HOUSED AT !WHICH!:  <NULWHS-N>; <NULPWHS>; <NULPWHS-PN>:
                      CORE HAS NONHUMFLG.
        HOUSED AT <NULWHS-N>:  <WHS-N>;
                  <NULPWHS>:  <PWHS>;
                  <NULPWHS-PN>:  <PWHS-PN>;
                  <WHS-N>, <PWHS>, <PWHS-PN>:  <RNWH>;
                  <RNWH>:  <RN>:
                    CORE HAS HUMFLG, NONHUMFLG OF CORE2.
        HOUSED AT <RN>:  <LNR>,1,2; <LNR>,2,2; <LPROOR>:
           $SS  OR  IF CORE2 HAS HUMFLG THEN CORE1 HAS PRO:PLURAL OR
                         NHUMAN OR AGGREGATE.
           $SS  OR  IF CORE2 HAS NONHUMFLG THEN CORE1 HASNT NHUMAN OR
                         PRO:PLURAL.
* WWH51:  "WHAT" IS NOT AN ACCEPTABLE STRING HEAD IN WH-N, PWH-PN, AND PWH
*     WHEN THESE STRINGS OCCUR IN RN OR RNSUBJ (*"THE BOOK WHAT I READ",
*     *"THE BOOK TO WHAT I REFERED").
WWH51:  HOUSED AT !WHAT!:  <NULWHS-N>; <NULPWHS-PN>; <NULPWHS>:
                      CORE HAS WHATFLG.
        HOUSED AT <NULWHS-N>:  <WHS-N>;
                  <NULPWHS-PN>:  <PWHS-PN>;
                  <NULPWHS>:  <PWHS>:
                    CORE HAS WHATFLG OF CORE2.
        HOUSED AT <WHS-N>, <PWHS-PN>, <PWHS>:  <RNWH>:
           $SS  OR  CORE HASNT WHATFLG.
* WWH52:  IN LNR OR VINGOFN TPOS TAKES THE VALUES "WHOSE", "WHICH", "WHAT"
*     ONLY IF TPOS IS OCCURRING IN A WH-QUESTION OR WH-STRING ("WHICH BOOK
*     DID YOU CHOOSE", "ON WHOSE BRANCHES THE LEAVES ARE FOUND").
WWH52:  HOUSED AT <WHLN>:  <TPOS>:
                      CORE HAS WHLNFLG.
        HOUSED AT <TPOS>:  <TQPOS>; <LN1>; <LN2>;
                  <TQPOS>:  <LNVAR>;
                  <LNVAR>:  <LNR>;
                  <LN1>:  <LNR>;
                  <LN2>:  <VINGOFN>:
                    CORE HAS WHLNFLG OF CORE2.
        HOUSED AT <TPOS>: ALL OCCURRENCES EXCEPT <TQPOS>; <LN1>; <LN2>;
                  <LNR>, <VINGOFN>:  ALL OCCURRENCES EXCEPT <WHN>:
           $POS3 OR CORE2 HASNT WHLNFLG.
        HOUSED AT <LNR>, <VINGOFN>:  <WHN>:
           $POS3 OR CORE HAS WHLNFLG.
* WWH53:  IN WHQ-N IF VERB IS HAVE, BE, DO THEN YESNOQ OPTION MUST BE USED.
WWH53:  HOUSED AT <VERB>:  <ASSERTIONVO>:
                      IF CORE2 HAS VHAVE OR VBE OR VDO THEN CORE HAS
                           VHAVEBEDOFLG.
        HOUSED AT <ASSERTIONVO>:  <ASSERTION>:
                      CORE HAS VHAVEBEDOFLG OF CORE2.
        HOUSED AT <ASSERTION>:  <NUL2WHQ-N>:
           $POS4 OR CORE2 HASNT VHAVEBEDOFLG.
* WWH54:  THE ASSERTION OPTION DOES NOT FOLLOW WHOM.
WWH54:  HOUSED AT <YESNOQ>:  <NUL2WHQ-N>;
                  !WHOM!:  <NUL1WHQ-N>;
                    CORE HAS WHOMFLG.
        HOUSED AT <NUL1WHQ-N>:  <WHQ-N>:
```

```
                CCRE HAS WHOMFLG OF CORE2.
HOUSED AT <NUL2WHQ=N>;  <WHQ=N>;
   $POS4 OR IF CORE1 HAS WHOMFLG THEN CORE2 HAS WHOMFLG.
```

APPENDIX C

INDEX OF FLAGS

THE FLAGS USED IN THE RESTRICTIONS OF APPENDIX B ARE LISTED HERE IN ALPHABETICAL ORDER. FOLLOWING EACH FLAG IS A BRIEF EXPLANATION OF ITS INTENDED MEANING AND A LIST OF THE RESTRICTIONS IN WHICH IT APPEARS.

ABOUTFLG: DP = P STRING HAS DP "ABOUT"
WPCS15

ACCFLG: ACCUSATIVE WH-WORD
WWH6A

ACROSSFLG: DP = P STRING HAS DP "ACROSS"
WPCS15

ADJFLG: SCALESTG HAS VALUE ADJ
DQ3

ADJINRNFLG: RN TAKES VALUE ADJINRN
WN50

ADJOMITFLG: PN RIGHT ADJUNCT OF NOUN HAS OMITTED NOUN STRING
WWH1C, WWH3B

ALONGFLG: DP = P STRING HAS DP "ALONG"
WPCS15

APOSFLG: ADJECTIVE POSITION OF LEFT ADJUNCT OF NOUN IS FILLED
WCOM2, DN1, DN50, WN10, WSN11

APPOSFLG: VALUE OF RN IS APPOS
WAGREE5, DCOM4, DN52, DN53

AROUNDFLG: DP = P STRING HAS DP "AROUND"
WPCS15

ASTGOBFLG: DEEPEST OBJECT OF BE TAKES VALUE ASTG
DVOB1, DVOB4, DVOB5, DPOS18, WSN2

AWAYFLG: DP = P STRING HAS DP "AWAY"
WPCS15

BACKFLG: DP = P STRING HAS DP "BACK"
WPCS15

C1SHOULDFLG: RN OR RNSUBJ TAKES VALUE C1SHOULD
DSN7

CENTERFLG: SA CAN ONLY APPEAR IN CENTER-LIKE STRINGS
DPCS7

```
DOBFLG:  CCRE IS DEEPEST OBJECT OF BE
         DVCB1, DVOB4, DVOB5, WAGREE2, WSEL2, WSN1, WSN2, WSN3, WSN8,
         WVC8

DOWNFLG:  DP = P STRING HAS DP "DOWN"
         WPCS15

DP4FLG:  OBJECT TAKES VALUE DP4
         DSN10, WSN11

DPFLG:  OBJECT OR PASSOBJ HAS DP STRING AS ITS VALUE
         WPCS15

DTPFLG:  RN TAKES VALUE DSTG, THATS-N, OR PN
         DN54

DVBFLG:  CCRE IS DEEPEST VERB
         DVCB2, DVOB4, DVOB5, WSEL2, WSN8

DVBVENPASSFLG:  DEEPEST VERB OCCURS IN PASSIVE VENPASS CONSTRUCTION
         DVCB4, WSEL2, WSN1

FORTOVOFLG:  SUBJECT OR OBJECT OF BE TAKES VALUE FORTOVO
         DSN1, WSN3

HOWQFLG:  TPOS HAS VALUE HOWQSTG OR HOWQASTG
         DN1

HUMFLG:  HUMAN WH-WORD
         WWH6B

IFFLG:  WHETHS BEGINS WITH "IF"
         WSN6

INDNAMFLG:  HOST OF THIS RIGHT ADJUNCT IS AN INDEFINITE PRONOUN OR
                  NON-NAME NOUN
         DN56

INFLG:  DP = P STRING HAS DP "IN"
         WPCS15

INTADJFLG:  THERE IS AN INTERVENING ADJUNCT
         DCOM2

INTRVFLG:  THERE IS AN INTERVENING RV
         WCOM4, WCOM7

INTSAFLG:  THERE IS AN INTERVENING SA
         WCOM4, WCOM5

LNVINGFLG:  PROPER LEFT ADJUNCT REQUIRED FOR VING IN NOUN PHRASE
         WN10
```

LPFLG: PREPOSITION HAS LEFT ADJUNCT
WCOM7, WPOS25

MOTIONFLG: GOVERNING VERB MUST BE VMOTION
WPOS2

NNFLG: OBJECT TAKES VALUE NN
WPOS22

NOAPOSFLG: CONSTRUCTION CANNOT APPEAR IN APOS
WQ1

NOMFLG: NOMINATIVE WH-WORD
WWH6A

NONHUMFLG: NON-HUMAN WH-WORD
WWH6B

NOPROFLG: CERTAIN SA'S CANNOT APPEAR IN PROSENT
WPOS19

NPOSFLG: NOUN POSITION OF LEFT ADJUNCT OF NOUN IS FILLED
WCOM2, WN10, WPOS21

NQFLG: QNPOS TAKES VALUE NQ
DN1

NSENTPFLG: SENTENTIAL NOUN WHICH MUST OCCUR IN PREPOSITIONAL PHRASE
DSN5

NSTARTFLG: OBJECT STARTS WITH NOUN AND IS VALID AFTER "OF" IN
VINGOFN
WSN11

NSTGOFLG: OBJECT TAKES VALUE NSTGO
WSEL1

NULLNFLG: NOUN PHRASE WITH ZEROED NOUN
DSN1

NULLOBJBEFLG: OBJECT OF BE OMITTED FROM STRING
WWH1C, WWH4

NULLOBJFLG: NULL OPTION TAKEN FOR OBJECT
WAGREE3, WPOS18, DVC1

NULLSUBJFLG: SUBJECT IS OMITTED IN WH-CONSTRUCTION
WAGREE6, DPOS3, WWH1B, WWH1C, WWH2

NULLWHNOFLG: NOUN STRING OBJECT OMITTED FROM STRING
WWH1A, WWH1B, WWH1C, WWH3A, WWH3B

NULLWHPOFLG: PREPOSITIONAL-PHRASE OBJECT OMITTED FROM STRING
WWH1B, WWH1C, WWH3B, WWH5

OFDP4FLG: "OF" OR DP4 OBJECT REQUIRED IN VINGOFN
WSN11

OFFFLG: DP = P STRING HAS DP "OFF"
WPOS15

OFFLG: "OF" OCCURS IN VINGOFN
WSN11, DVC1

OKADJINRNFLG: VALID ADJECTIVE CONSTRUCTION FOR RIGHT ADJUNCT OF NOUN
WN50, DQ1

OKC1SHOULDFLG: SUBJECT IS SOBJBE COMPATIBLE WITH C1SHOULD IN RNSUBJ
DSN7

OKCOUNTFLG: VALID LEFT ADJUNCT FOR COUNT NOUN
WN9

ONFLG: DP = P STRING HAS DP "ON"
WPOS15

ONLYAPOSFLG: CONSTRUCTION CAN OCCUR ONLY IN APOS
WQ1

ONLYOBFLG: CONSTRUCTION CAN ONLY OCCUR AS AN OBJECT OF BE
DN52, DN54, WN9

ONLYPNFLG: CONSTRUCTION CAN ONLY OCCUR IN PREPOSITIONAL PHRASE
WN9

OMITFLG: OBJECT OMITTED FROM WH-STRING
WWH1A, WWH1B, WWH1C, WWH3A, WWH3B, WWH6A

OMITR1FLG: FACT OF OBJECT OMISSION FROM WH-STRING TO BE PASSED UP
THROUGH RIGHT ADJUNCTS
WWH1A, WWH1C

OMITR2FLG: AND FARTHER UP THROUGH RIGHT ADJUNCTS OF NOUNS
WWH1A, WWH1C

ORNOTFLG: "OR NOT" APPEARS IN YESNOQ OR WHETHS
DPOS11, WSN7

OUTFLG: DP = P STRING HAS DP "OUT"
WPOS15

OVERFLG: DP = P STRING HAS DP "OVER"
WPOS15

PLURALFLG: CORE IS PLURAL

DVCB1, DVOB4, DVCB5, WAGREE1, WAGREE2, WAGREE4, WAGREE5, WAGREE6, WAGREE7, WQ1

PNFLG: RN, RV, OR SCALESTG TAKES VALUE PN
WN51, WPOS18, DQ2, DQ3

POBJFLG: OBJECT OR PASSOBJ HAS PREPOSITION CONSTRUCTION AS ITS VALUE
WPOS15

POMITFLG: PREPOSITIONAL PHRASE OMITTED FROM WH-STRING
WWH1A, WWH1B, WWH1C, WWH5

POMITR1FLG: FACT OF PREPOSITIONAL PHRASE OMISSION FROM WH-STRING TO BE PASSED UP THROUGH RIGHT ADJUNCTS
WWH1A, WWH1C

POMITR2FLG: AND FARTHER UP THROUGH RIGHT ADJUNCTS OF NOUNS
WWH1A, WWH1C

POSTOBJFLG: PN WITH OMITTED NOUN STRING CANNOT OCCUR IN POST-OBJECT POSITION OF STRING
WWH1C, WWH3B

PRCACCFLG: SUBJECT OF ASSERTION IS ACCUSATIVE PRONOUN
WPOS5

PRONOMFLG: SUBJECT OF ASSERTION IS NOMINATIVE PRONOUN
WPOS5

QNPOSFLG: LP TAKES VALUE QNPOS
WQ7

QPOSFLG: QUANTIFIER POSITION OF LEFT ADJUNCT OF NOUN IS FILLED
WCOM2, DN1, DN50, DN51, WN10, WSN11

QUESTIONFLG: CENTER IS QUESTION
WPOS23

RATOVOFLG: RA TAKES VALUE TOVO
DSN9

RNFLG: PRONOUN HAS RIGHT ADJUNCT
WPOS8

RNSNFLG: RN HAS VALUE SN
DSN4, DSN5

RNSUBJFLG: SA HAS VALUE RNSUBJ
DPOS3, DPOS4

RSPLURFLG: POST-OBJECT RIGHT ADJUNCT OF SUBJECT IS PLURAL
WAGREE8

RSSINGFLG: POST-OBJECT RIGHT ADJUNCT OF SUBJECT IS SINGULAR
WAGREE7

S-NFLG: RN TAKES VALUE S-N
DN50, DN51

SAFLG: LVSA APPEARS IN VENPASS
DPCS10

SAINNERFLG: SA CANNCT APPEAR IN INITIAL POSITION
DPCS2, DPOS12

SCALEFLG: NSTGO IN PN HAS ATTRIBUTE NSCALE
WQ3

SHCRTSAFLG: SA IS SHORT
WCCM4, WCOM5

SHCULDFLG: C1SHOULD CONTAINS WORD "SHOULD"
WSN4

SINGFLG: CORE IS SINGULAR
DVCB1, DVOB4, DVOB5, WAGREE1, WAGREE2, WAGREE3, WAGREE4, WAGREE5,
WAGREE7, WAGREE8

SNACBFLG: OBJBE TAKES VALUE SN OR ASSERTION
DSN1, WSN3

SNARAFLG: RA TAKES VALUE SN OR ASSERTION
DSN2

SOMITFLG: SUBJECT CMITTED FROM WH-STRING
WWF1A, WWH1B, WWH1C, WWH2, WWH4, WWH6A

SOMITR1FLG: FACT OF SUBJECT OMISSION FROM WH-STRING TO BE PASSED UP
THRCUGH RIGHT ADJUNCTS
WWF1A, WWH1C

SOMITR2FLG: AND FARTHER UP THROUGH RIGHT ADJUNCTS OF NOUNS
WWF1A, WWH1C

STGFLG: NSTG OR WHN HAS STRING AS VALUE
WAGREE8

SUBJITFLG: ULTIMATE SUBJECT REQUIRED TO BE "IT"
DSN2, DSN5

SUBJSNFLG: SUBJECT HAS VALUE SN
DSN2, DSN8, WSN1, WSN2

SUBJVINGFLG: SUBJECT HAS VALUE VINGSTG
DSN2

SUBJWHFLG: SUBJECT HAS AS VALUE NSTG BEGINNING WITH WH-WORD
DSN2

THANFLG: SAWHICHSTG BEGINS WITH "THAN"
DPCS5

THATFLG: C1SHOULD BEGINS WITH "THAT"
WSN5, WWH2

THRCUGHFLG: DP - P STRING HAS DP "THROUGH"
WPCS15

TIMETAGFLG: RIGHT CR LEFT ADJUNCT OF NOUN HAS TIMETAG
WPCS10

TOVC-NFLG: RA TAKES VALUE TOVO-N
DSN6

TOVCFLG: SUBJECT OR OBJECT OF BE TAKES SN VALUE TOVO
DSN1, WSN3

TOVCOBFLG: OBJECT OF BE HAS NON-SN VALUE TOVO
DPCS1

TPCSFLG: ARTICLE POSITION OF LEFT ADJUNCT OF NOUN IS FILLED
WCCM2, DN1, DN50, DN51

TVFLG: VERB TAKES VALUE TV
DSN11

UPFLG: DP - P STRING HAS DP "UP"
WPCS15

VBOBJFLG: VALUE OF OBJECT IS VERBAL-OBJECT STRING
DVCB3, DVOB4

VENCFLG: OBJECT TAKES VALUE VEND
WVC3

VENPASSFLG: RN TAKES VALUE VENPASS
WSEL5

VERB2FLG: VERB2 OCCURS IN YESNOQVC OR TSUBJVOVO
DVC1, DVC3

VFLG: VERB TAKES VALUE LVR
DSN11

VHAVEBEDOFLG: VERB IS VHAVE OR VBE OR VDO
WWH53

VMODFLG: GOVERNING VERB MUST BE VMOD
WPCS22

```
WFLG:  VERB1 HAS VALUE LWR
       DVC3

WHATFLG:  WH-STRING STARTS WITH "WHAT"
       WWH51

WHENFLG:  WHENS BEGINS WITH "WHEN"
       WCCM8

WHENSFLG:  RNWH TAKES VALUE WHENS
       DPCS14

WHICHFLG:  SAWHICHSTG BEGINS WITH  "WHICH"
       DPCS5, WSN8

WHLNFLG:  WH- LEFT ADJUNCT OF NOUN
       WWF52

WHOMFLG:  QUESTION BEGINS WITH "WHOM"
       WWF54

WVFLG:  VERB TAKES VALUE LWR - LVR
       DSN11
```

```
*
*                          APPENDIX  D
*
*              STRING GRAMMAR OF ENGLISH USED IN CHAPTER 4
*                FOR STATISTICAL TESTS OF BNF COMPONENT
*
*
* 1. SENTENCE
*
<SENTENCE>        ::= [<INTRODUCER>] <CENTER> <ENDMARK>  .                    1.1
<INTRODUCER>      ::= AND / OR / BUT / FOR .                                  1.2
<CENTER>          ::= <CENASR> / <QUESTION> /<PROSENT>/<PERMUTATION>.         1.3
<ENDMARK>         ::= !.! / ! ! / !;!  .                                      1.4
*
* 2. CENTER STRINGS
*
<CENASR>          ::= [<SA>] <SUBJECT> [<SA>] <VERB> [<SA>] [<OBJECT>]        2.1
                      [<RV>] [<SA>] .
<ASSERTION>       ::= [<SA>] [<SUBJECT>] [<SA>] <VERB> [<SA>] [<OBJECT>]      2.1A
                      [<RV>] [<SA>] .
<QUESTION>        ::= <YESNOQ> / <WHQ-N> / <WHQ> / <PWHQ-PN> /                2.2
                      <PWHQ> / <WHNQ-N> / <PWHNQ-PN> / <PWHNQ>  .
<PROSENT>         ::= THIS <SA> .                                             2.3
<PERMUTATION>     ::= <TSUBJVO> / <OBES> .                                    2.4
*
* 3.  QUESTION CENTERS
*
<YESNOQ>          ::= [<SA>] <VERB1> <SUBJECT> [<RW>] [<SA>] [<VERB2>]       3.1
                      [<OBJECT>] [<RV>] [<SA>] [<ORNOT>] .
<ORNOT>           ::= OR NOT .                                                3.2
<WHQ-N>           ::= <NUL1WHQ-N> <NUL2WHQ-N>  .                              3.3
<NUL1WHQ-N>       ::= WHO / WHOM / WHICH / WHAT / WHOSE .                     3.3A
<NUL2WHQ-N>       ::= <YESNOQ> / <ASSERTION> .                                3.3B
<WHQ>             ::= [<SA>] <NULWHQ> <YESNOQ> .                              3.4
<NULWHQ>          ::= WHERE / WHEN / HOW / WHY .                              3.4A
<PWHQ-PN>         ::= [<SA>] <*P> <NULPWHQ> <YESNOQ> .                        3.5
<PWHQ>            ::= [<SA>] <*P> <NULPWHQ> <YESNOQ> .                        3.6
<NULPWHQ>         ::= WHOM / WHICH / WHAT / WHOSE .                           3.6A
<WHNQ-N>          ::= [<SA>] <NULWHNQ> <YESNOQ> .                             3.7
<PWHNQ-PN>        ::= [<SA>] <*P> <NULWHNQ> <YESNOQ> .                        3.8
<PWHNQ>           ::= [<SA>] <*P> <NULWHNQ> <YESNOQ> .                        3.9
<NULWHNQ>         ::= <LNR> / <VINGOFN> .                                     3.9A
*
* 4.  PERMUTATION CENTERS
*
<TSUBJVO>         ::= <NULTSUBJVO> [<SA>] <VERB1> <SUBJECT> [<SA>]            4.1
                      [<VERB2>] [<OBJECT>] [<RV>] [<SA>]  .
<NULTSUBJVO>      ::= NEITHER / NOR / <DSTG> / <PN>  .                        4.1A
<OBES>            ::= <NULOBES> [<SA>] <VERB3> [<VERB4>] [<SA>] <SUBJECT>     4.2
                      [<SA>]  .
<NULOBES>         ::= <PN> / <ASTG> / <VENPASS> / <DSTG> .                    4.2A
*
* 5.  SUBJECT STRINGS
```

```
*
<SUBJECT>          ::= <NSTG> / <VINGSTG> / <SN> .                                  5.1
<NSTG>             ::= <LNR> / <NWHSTG> / <NAMESTG> / <LPRCR> .                     5.2
<LNR>              ::= [<LN>] <NVAR> [<RN>] / <LN> [<RN>] .                        5.3
<NVAR>             ::= <*N> / <*VING> .                                            5.4
<NWHSTG>           ::= <WHATS-N> / <WHERES> / <WHEVERS-N> .                        5.5
<NAMESTG>          ::= <*N> <NULNAMESTG> .                                         5.6
<NULNAMESTG>       ::= <NUMBRSTG> / <*N> .                                         5.6A
<LPRCR>            ::= [<LPRC>] <*PRO> [<RN>] .                                    5.7
<VINGSTG>          ::= <NSVINGO> / <VINGOFN> .                                     5.8
<NSVINGO>          ::= [<TPOS>] <VINGO> .                                          5.9
<VINGOFN>          ::= [<LN>] <*VING> [<NUL1VINGOFN>] [<RV>] [<SA>] .              5.10
<NUL1VINGOFN>      ::= <OBJECT> / OF [<NUL2VINGOFN>] .                             5.10A
<NUL2VINGOFN>      ::= <NSTGO> / <VINGSTG> / <NPN> .                               5.10B
<SN>               ::= <THATS> / <FORTOVO> / <TOVO> / <SVINGO> /                   5.11
                       <CISHOULD> / <SNWH> / <THATVING> .
*
* 6.  SENTENCE NOMINALIZATION STRINGS SN
*
<THATS>            ::= THAT <ASSERTION> .                                          6.1
<FORTOVO>          ::= FOR <SUBJECT> [<SA>] <TOVO> .                               6.2
<SVINGO>           ::= <NULSVINGO> <VINGO> .                                       6.3
<NULSVINGO>        ::= <NSTG> / <VINGSTG> .                                        6.3A
<THATVING>         ::= <SN> <VINGO> .                                              6.4
<CISHOULD>         ::= [THAT] <ASSERTION> .                                        6.5
<SNWH>             ::= <WHETHS> / <WHETHTOVO> / <WHS-N> / <PWHS> /                 6.6
                       <PWHS-PN> / <WHNS-N> / <PWHNS> / <PWHNS-PN> .
*
* 7.  VERB AND VERBAL OBJECT STRINGS
*
<VERB>             ::= <LTVR> / <LWR> [<SA>] <LVR> .                               7.1
<LTVR>             ::= [<LV>] <*TV> [<RW>] [<RV>] .                                7.2
<LWR>              ::= [<LW>] <*W> [<RW>] .                                        7.3
<LVR>              ::= [<LV>] <*V> [<RV>] .                                        7.4
<VENO>             ::= <LVENR> [<SA>] [<OBJECT>] [<RV>] .                          7.5
<LVENR>            ::= [<LV>] <*VEN> [<RV>] .                                      7.6
<VENPASS>          ::= [<LVSA>] <LVENR> [<SA>] [<PASSOBJ>] [<RV>] [<SA>] .         7.7
<LVSA>             ::= <DSTG> .                                                    7.8
<VINGO>            ::= [<LVSA>] <LVINGR> [<SA>] [<OBJECT>] [<RV>] [<SA>] .         7.9
<LVINGR>           ::= [<LV>] <*VING> [<RV>] .                                     7.10
<VO>               ::= <LVR> [<SA>] [<OBJECT>] [<RV>] .                            7.11
<TOVO>             ::= [<LV>] TO <LVR> [<SA>] [<OBJECT>] [<RV>] [<SA>] .           7.12
<VERB1>            ::= <LWR> / <LTVR> .                                            7.13
<VERB2>            ::= <LVR> [<SA>] .                                              7.14
<VERB3>            ::= <LTVR> / <*W> [<RW>] [<SA>] <LVR> .                         7.15
<VERB4>            ::= <*VEN> / TO <*V> .                                          7.16
*
* 8.  OBJECT STRINGS
```

```
*
<OBJECT>          ::= <THATS> / <NSTG> / <TOVO> / <PN> / <PVINGSTG> /      8.1
                      <PVING> / <PSNWH> / <PNN> / <PNVINGSTG> /
                      <PNTHATS> / <PNSNWH> / <PNTHATSVO> / <PSVING> /
                      <DP2> / <DP4> / <DP2PN> / <DP4PN> / <DPSN> /
                      <DP1PN> / <DP1> / <OBJECTBE>  / <VENO> /
                      <VO>  /   <SASOBJBE> / <NPN> / <NANDN> /
                      <VINGSTGPN> / <NPVINGSTG> / <ASOBJBE> / <NPSVINGO>
                      / <NPVINGO> / <NPSNWH> / <DP3> / <OP3PN> /
                      <OBJBE> / <VINGSTG> / <SNWH> / <WHETHS> /
                      <SVING> / <C1SHOULD> / <FORTOVO> / <NTOVO> /
                      <NN> / <SOBJBE> / <SVEN> / <NA> / <ADJN> / <ND>
                      / <VINGO> / <SVO> / <ASTG> / <NTHATS>
                      / <DSTG> / <NSNWH> / <ASSERTION> / <STOVO-N> .
<PASSOBJ>         ::= <THATS> / <PN> / <PVINGSTG> / <PSVINGO> /          8.2
                      <PVING> / <PSNWH> / <ASOBJBE> /
                      <DP1> / <P1> / <DP1P> / <DP1PN> /
                      <VINGO> / <FORTOVO> / <TOVO> / <ASSERTION> /
                      <NSTGO> / <SNWH> / <OBJBE> / <VENPASS> / <ASTG>
                      / <DSTG> / <VO> .
<OBJECTBE>        ::= <VENPASS> / <VING> / <OBJBE> / <SN> / <VINGSTG>    8.3
                      / <TOVO> / <ASSERTION> / <BEINGO> .
<OBJBE>           ::= <ASTG> / <NSTG> / <PN> / <DSTG> .                  8.4
<BEINGO>          ::= [<LV>] BEING <NULBEINGO> .                         8.5
<NULBEINGO>       ::= <LAR> / <NSTG> .                                   8.5A
<NSTGO>           ::= <NSTG> .                                           8.6
<NN>              ::= [<NSTGO>] [<SA>] [<NSTGO>] .                       8.7
<NANDN>           ::= [<NULNANDN>] AND [<NULNANDN>] .                    8.7A
<NULNANDN>        ::= <NSTGO> / <VINGSTG> .                              8.7B
<ASTG>            ::= <LAR> / <LQAR> .                                   8.8
<LAR>             ::= [<LA>] <AVAR> [<RA>] .                             8.9
*
* 8A. P STRINGS
*
<P1>              ::= <*P> .                                             8A.1
<PSTG>            ::= <PN> / <PVINGSTG> / <PSVINGO> .                    8A.2
<PN>              ::= [<LP>] <*P> [<NSTGO>] .                            8A.3
<PVINGSTG>        ::= <*P> <VINGSTG> .                                   8A.4
<PSVINGO>         ::= <*P> <SVINGO> .                                    8A.5
<PVINGO>          ::= <*P> <VINGO> .                                     8A.6
<PSNWH>           ::= <*P> <SNWH> .                                      8A.7
<NPN>             ::= [<NSTGO>] [<SA>] [<PN>] .                          8A.8
<PNN>             ::= [<PN>] [<SA>] [<NSTGO>] .                          8A.9
<VINGSTGPN>       ::= <VINGSTG> [<SA>] [<NULVINGSTGPN>] .                8A.10
<NULVINGSTGPN>    ::= <PN> / <PVINGSTG> / <PSVINGO> / <PVINGO> .         8A.10A
<PNVINGSTG>       ::= [<NULVINGSTGPN>] [<SA>] <VINGSTG> .                8A.11
<NPVINGSTG>       ::= [<NSTGO>] [<SA>] <PVINGSTG> .                      8A.12
<NPSVINGO>        ::= [<NSTGO>] [<SA>] <PSVINGO> .                       8A.13
<NPVINGO>         ::= [<NSTGO>] [<SA>] <PVINGO> .                        8A.14
<NPSNWH>          ::= [<NSTGO>] [<SA>] <*P> <SNWH> .                     8A.15
*
* 8B.  DP STRINGS
*
```

```
<DP1>           ::= <*DP> .                                                  8B.1
<DP2>           ::= <*DP> [<NULNANDN>] .                                     8B.2
<DP3>           ::= [<NULNANDN>] <*DP> .                                     8B.3
<DP4>           ::= <*DP> OF [<NULNANDN>] .                                  8B.4
<DPSN>          ::= <*DP> [<SA>] <SN> .                                      8B.5
<DP1PN>         ::= <*DP> [<SA>] [<NULDPPN>] .                               8B.6
<NULDPPN>       ::= <PN> / <PVINGSTG> / <PSVINGO> .                          8B.6A
<DP2PN>         ::= <DP2> [<NULDPPN>] .                                      8B.7
<DP3PN>         ::= <DP3> [<NULDPPN>] .                                      8B.8
<DP4PN>         ::= <DP4> [<NULDPPN>] .                                      8B.9
<DP1P>          ::= <*DP> <*P> .                                             8B.10
*
* 8C.  OBJECT NOMINALIZATION STRINGS
*
<NTOVC>         ::= [<SUBJECT>] [<RV>] [<SA>] <TOVO> .                       8C.1
<SVO>           ::= <ASSERTION> .                                            8C.2
<NTHATS>        ::= [<NSTGO>] [<RV>] [<SA>] <NULNTHATS> .                    8C.3
<NULNTHATS>     ::= <THATS> / <ASSERTION> .                                  8C.3A
<NSNWH>         ::= [<NSTGO>] [<RV>] [<SA>] <SNWH> .                         8C.4
<PNTHATS>       ::= <*P> [<NSTGO>] [<RV>] [<SA>] <NULNTHATS> .               8C.5
<PNTHATSVC>     ::= [<PN>] [<RV>] [<SA>] <C1SHOULD> .                        8C.6
<PNSNWH>        ::= [<PN>] [<RV>] [<SA>] <SNWH> .                            8C.7
<STOVC-N>       ::= [<SUBJECT>] [<SA>] <TOVO-N> .                            8C.8
*
* 8D.  NOMINALIZATIONS WITH ZEROED VERB BE
*
<SOBJBE>        ::= [<SUBJECT>] [<SA>] [<OBJBE>] [<SA>] .                    8D.1
<SVEN>          ::= [<SUBJECT>] [<SA>] <VENPASS> [<SA>] .                    8D.2
<NA>            ::= [<NSTGO>] [<SA>] <LAR> .                                 8D.3
<ADJN>          ::= <ASTG> [<SA>] [<NSTGO>] .                                8D.4
<ND>            ::= [<NSTGO>] [<SA>] <DSTG> .                                8D.5
<SASOBJBE>      ::= [<SUBJECT>] [<SA>] AS  [<NULSASOBJBE>] [<SA>] .          8D.6
<NULSASOBJBE>   ::= <OBJBE> / <VINGC> .                                      8D.6A
<ASOBJBE>       ::= AS [<OBJBE>] .                                           8D.7
*
* 9.  SENTENCE ADJUNCT STRINGS
*
<SA>            ::= [<SA>] [<REPSA>] .                                       9.1
<REPSA>         ::= <*INT>/ <DSTG> [<RD>] / <PN> / <PA> / <NSTGT>/           9.1A
                    <RSUBJ> / <RNSUBJ> / [<LCS>] <CSSTG> / <OBJBESA> /
                    <SOBJBESA> / <VINGO> / <VENPASS> / <SAWH> /
                    <TOVC> / <NVSA> .
<DSTG>          ::= <*D> / <DSTG> <*D> .                                     9.2
<PA>            ::= <*P> <*ADJ> .                                            9.3
<NSTGT>         ::= <LNR> .                                                  9.4
<RSUBJ>         ::= <NUL1RSUBJ> [<NUL2RSUBJ>] / <*PRO> .                     9.5
<NUL1RSUBJ>     ::= <*Q> / <*T> .                                            9.5A
<NUL2RSUBJ>     ::= <PN> / <*D> .                                            9.5B
<RNSUBJ>        ::= <SN> / <RNWH> / <THATS-N> .                              9.6
<CSSTG>         ::= <SUB1> / <SUB2> / <SUB3 > / <SUB0> / <SUB5> /            9.7
                    <SUB6> / <SUB4> / <SUB8> .
<OBJBESA>       ::= [<OBJBE>] [<SA>] .                                       9.8
<SOBJBESA>      ::= <NSTG> [<SA>] [<NULSOBJBESA>] .                          9.9
```

```
<NULSOBJBESA>     ::= <OBJBESA> / <VENPASS> / <VINGO> .                          9.9A
<SAWH>            ::= <SAWHICHSTG> / <WHETHS> / <WHEVERS-N> .                    9.10
<NVSA>            ::= <NSTG> [<SA>] <VERB> .                                     9.11
*
* 10.  SUBORDINATE CONJUNCTION (CS) STRINGS
*
<SUB0>            ::= <*CS0> [<OBJBE>] [<SA>] .                                 10.1
<SUB1>            ::= <*CS1> <ASSERTION> .                                      10.2
<SUB2>            ::= <NULSUB2> <VENPASS> .                                     10.3
<NULSUB2>         ::= <*CS2> / AS / THAN .                                      10.3A
<SUB3>            ::= <*CS3> <VINGO> .                                          10.4
<SUB4>            ::= <*CS4> <VINGSTG> .                                        10.5
<SUB5>            ::= <*CS5> <SVINGO> .                                         10.6
<SUB6>            ::= <*CS6> <SOBJBE> .                                         10.7
<SUB7>            ::= <*CS7> <SVEN> .                                           10.8
<SUB8>            ::= <NULSUB8> <*D> .                                          10.9
<NULSUB8>         ::= AS / THAN .                                               10.9A
*
* 11.  RN RIGHT ADJUNCTS OF N
*
<RN>              ::= [<RN>] [<REPRN>] .                                        11.1
<REPRN>           ::= <VENPASS> / <RNP> / <RNWH> / <THATS-N> / <S-N> /          11.1A
                      <SN> / <VINGO> / <TOVO> / <TOVO-N> / <FORTOVO-N>/
                      <ADJINRN> / <OSTG> / <APPOS> / WHATSOEVER .
<RNP>             ::= <PN> / <PVINGSTG> / <PSVINGO> / <PSNWH> .                 11.2
<RNWH>            ::= <WHS-N> / <WHENS> / <PWHS> / <PWHS-PN> /                  11.3
                      <WHNS-N>   / <PWHNS-PN> / <PWHNS> .
<TOVO-N>          ::= <TOVO> .                                                  11.4
<FORTOVO-N>       ::= <FORTOVO> .                                               11.5
<ADJINRN>         ::= <LAR> / <LQNR> .                                          11.6
<APPOS>           ::= <NSTG> .                                                  11.7
*
* 12.  LN LEFT ADJUNCTS OF N
*
<LN>              ::= [<TPOS>] [<QPOS>] [<APOS>] [<NSPOS>] [<NPOS>] .           12.1
<TPOS>            ::= <LTR> / <WHLN> / <LNSR> / <HOWQSTG> .                     12.2
<LTR>             ::= [<LT>] [<*T>] .                                           12.3
<LNSR>            ::= [<LN>] <*NS> .                                            12.4
<WHLN>            ::= WHOSE / WHICH / WHAT / <HOWQASTG> .                       12.5
<HOWQASTG>        ::= HOW <NULHOWQASTG> [OF] <*T> .                             12.6
<NULHOWQASTG>     ::= MUCH / MANY / <*ADJ> .                                    12.6A
<HOWQSTG>         ::= HOW <NULHOWQSTG> .                                        12.7
<NULHOWQSTG>      ::= MUCH / MANY .                                             12.7A
<QPOS>            ::= <LQR> .                                                   12.8
<LQR>             ::= [<LQ>] <QVAR> [<RQ>] .                                    12.9
<QVAR>            ::= <*Q> / <CPDNUMBR> .                                       12.10
<CPDNUMBR>        ::= <*Q> <*Q> / <CPDNUMBR> <*Q> .                             12.11
<APOS>            ::= <LAR1> / <LQNR> .                                         12.12
<LAR1>            ::= [<LA>] <AVAR> [<RA1>] .                                   12.13
<AVAR>            ::= [<LCDA>] <*ADJ> / [<LCDVA>] <NULAVAR> .                   12.14
<NULAVAR>         ::= <*VING> / <*VEN> .                                        12.14A
<LQNR>            ::= [<LQ>] <QNPOS> [<RQ>] .                                   12.15
<QNPOS>           ::= <QN> / <NQ> .                                             12.16
```

```
<QN>            ::= <NUMBRSTG> <*N> [<SCALESTG>] .                    12.17
<NUMBRSTG>      ::= <*Q> / <CPDNUMBR> .                               12.18
<SCALESTG>      ::= <*ADJ> / <PN> / OR SO .                           12.19
<NQ>            ::= <*N> <NUMBRSTG> .                                 12.20
<NSPOS>         ::= <*NS> .                                           12.21
<NPOS>          ::= [<LCDN>] <NULNPOS> .                              12.22
<NULNPOS>       ::= <*N> / <*VING> .                                  12.22A
*
* 13.  RIGHT ADJUNCTS  -  OTHER THAN RN
*
<RW>            ::= <DSTG> .                                          13.1
<RQ>            ::= OR SO / ENOUGH .                                  13.2
<RA>            ::= <SN> / <CENASR> / <PSTG> / <TOVO-N> /             13.3
                    <FORTOVO-N> / <TOVO> / ENOUGH .
<RA1>           ::= ENOUGH .                                          13.4
<RV>            ::= [<RV>] [<REPRV>] .                                13.5
<REPRV>         ::= <DSTG> [<RD>] / <PN> / <QN> .                     13.5A
<RD>            ::= ENOUGH .                                          13.6
*
* 14.  LEFT ADJUNCTS  --  OTHER THAN LN
*
<LW>            ::= <*D> .                                            14.1
<LV>            ::= <DSTG> .                                          14.2
<LQ>            ::= <DSTG> / [VERY] <NULLQ> .                         14.3
<NULLQ>         ::= <*ADJ> / <*VING> / <*VEN> .                       14.3A
<LA>            ::= <DSTG> .                                          14.4
<LP>            ::= <DSTG> / <QNPOS> .                                14.5
<LCS>           ::= <*D> .                                            14.6
<LPRO>          ::= <DSTG> .                                          14.7
<LT>            ::= <*D> /  <*Q>  .                                   14.8
<LCDA>          ::= <*N>  .                                           14.9
<LCDVA>         ::= <*N> / <*D> / <*ADJ> .                            14.10
<LCDN>          ::= <*ADJ> / <*Q> .                                   14.11
*
* 15.  WH-STRINGS
*
<WHETHS>        ::= <NULWHETHS> [<ORNOT>] <ASSERTION> [<ORNOT>] .     15.1
<NULWHETHS>     ::= WHETHER / WHERE / WHEN / HOW / WHY / IF .          15.1A
<WHETHTOVC>     ::= <NULWHETHS> <TOVO> .                              15.2
<WHS-N>         ::= <NULWHS-N> <ASSERTION> .                          15.3
<NULWHS-N>      ::= WHO / WHOM / WHICH / WHAT / WHOSE / THAN / AS .    15.3A
<PWHS>          ::= <*P> <NULPWHS> <CENASR> .                         15.4
<NULPWHS>       ::= WHOM / WHICH / WHAT / WHOSE / WHERE / WHEN .       15.4A
<PWHS-PN>       ::= <*P> <NULPWHS-PN> <CENASR> .                      15.5
<NULPWHS-PN>    ::= WHOM / WHICH / WHAT / WHOSE .                     15.5A
<WHNS-N>        ::= <NULWHNS-N> <ASSERTION> .                         15.6
<NULWHNS-N>     ::= <LNR> / <VINGCFN> .                               15.6A
<PWHNS>         ::= <*P> <NULWHNS-N> <ASSERTION> .                    15.7
<PWHNS-PN>      ::= <*P> <NULWHNS-N> <CENASR>                         15.8
<WHATS-N>       ::= WHAT <ASSERTION> .                                15.9
<WHERES>        ::= WHERE <CENASR> .                                  15.10
<WHENS>         ::= [<NULWHENS>] <CENASR> .                           15.11
<NULWHENS>      ::= WHERE / WHEN .                                    15.11A
```

```
<THATS-N>          ::= THAT <ASSERTION> .                                    15.12
<S-N>              ::= <CENASF> .                                            15.13
<SAWHICHSTG>       ::= <NULSAWHICHSTG> <ASSERTION> .                         15.14
<NULSAWHICHSTG>    ::= WHICH / THAN / AS .                                   15.14A
<WHEVERS-N>        ::= <NULWHEVERS-N> <ASSERTION> .                          15.15
<NULWHEVERS-N>     ::= WHOEVER / WHOMEVER / WHICHEVER / WHATEVER /           15.15A
                       WHOSCEVER .
```

DATE DUE

GAYLORD PRINTED IN U.S.A.